

SDHLibrary-CPP

0.0.2.10

Generated by Doxygen 1.7.3-20110123

Tue Sep 30 2014 15:59:05

Contents

1	SDHLibrary_CPP	1
1.1	General project information	1
1.2	Purpose	1
1.3	Links	1
1.4	Copyright	2
2	Bug List	3
3	Module Index	9
3.1	Modules	9
4	Namespace Index	11
4.1	Namespace List	11
5	Class Index	13
5.1	Class Hierarchy	13
6	Class Index	15
6.1	Class List	15
7	File Index	19
7.1	File List	19
8	Module Documentation	23
8.1	Compile time settings	23
8.1.1	Detailed Description	23
8.1.2	Define Documentation	23
8.1.2.1	SDH_USE_BINARY_COMMUNICATION	23
8.1.2.2	SDH_USE_NAMESPACE	23
8.2	Derived compile time settings	24
8.2.1	Detailed Description	24
8.2.2	Define Documentation	24
8.2.2.1	NAMESPACE_SDH_END	24
8.2.2.2	NAMESPACE_SDH_START	24
8.2.2.3	NS_SDH	24
8.2.2.4	USING_NAMESPACE_SDH	24
8.3	Demonstration programs	24
8.3.1	Detailed Description	26
8.4	Online help of demonstration programs	26

9 Namespace Documentation	55
9.1 SDH Namespace Reference	55
9.1.1 Detailed Description	60
9.1.2 Typedef Documentation	60
9.1.2.1 Int16	60
9.1.2.2 Int32	60
9.1.2.3 Int8	60
9.1.2.4 PCAN_HANDLE	60
9.1.2.5 pDoubleUnitConverterFunction	61
9.1.2.6 pGetFunction	61
9.1.2.7 pSetFunction	61
9.1.2.8 tCRCValue	61
9.1.2.9 tDeviceHandle	61
9.1.2.10 tTimevalSec	61
9.1.2.11 tTimevalUSec	61
9.1.2.12 UInt16	61
9.1.2.13 UInt32	61
9.1.2.14 UInt8	61
9.1.3 Enumeration Type Documentation	62
9.1.3.1 "@2	62
9.1.4 Function Documentation	62
9.1.4.1 __attribute__	62
9.1.4.2 apply	62
9.1.4.3 apply	62
9.1.4.4 Approx	62
9.1.4.5 Approx	63
9.1.4.6 CompareReleases	63
9.1.4.7 DegToRad	63
9.1.4.8 InIndex	63
9.1.4.9 InRange	63
9.1.4.10 InRange	63
9.1.4.11 map	64
9.1.4.12 NumerifyRelease	64
9.1.4.13 operator<<	64
9.1.4.14 operator<<	64
9.1.4.15 operator<<	64
9.1.4.16 operator<<	65
9.1.4.17 operator<<	65
9.1.4.18 operator<<	65
9.1.4.19 operator<<	65
9.1.4.20 operator<<	65
9.1.4.21 operator<<	65
9.1.4.22 operator<<	66
9.1.4.23 operator<<	66
9.1.4.24 operator<<	66
9.1.4.25 RadToDeg	66
9.1.4.26 SDHCommandCodeToString	66
9.1.4.27 SDHReturnCodeToString	66
9.1.4.28 SleepSec	66
9.1.4.29 ToRange	66

9.1.4.30	ToRange	66
9.1.4.31	ToRange	66
9.1.4.32	ToRange	66
9.1.5	Variable Documentation	67
9.1.5.1	g_sdh_debug_log	67
9.1.5.2	SDH__attribute__	67
9.1.5.3	uc_identity	67
10	Class Documentation	69
10.1	SDH::cCANSerial_ESD Class Reference	69
10.1.1	Detailed Description	73
10.1.2	Constructor & Destructor Documentation	73
10.1.2.1	cCANSerial_ESD	73
10.1.2.2	cCANSerial_ESD	73
10.1.2.3	~cCANSerial_ESD	74
10.1.3	Member Function Documentation	74
10.1.3.1	BaudrateToBaudrateCode	74
10.1.3.2	Close	74
10.1.3.3	GetErrorMessage	74
10.1.3.4	GetErrorNumber	74
10.1.3.5	GetHandle	74
10.1.3.6	IsOpen	74
10.1.3.7	Open	75
10.1.3.8	Read	75
10.1.3.9	SetTimeout	75
10.1.3.10	write	75
10.1.4	Member Data Documentation	75
10.1.4.1	baudrate	75
10.1.4.2	id_read	76
10.1.4.3	id_write	76
10.1.4.4	net	76
10.1.4.5	status	76
10.2	SDH::cCANSerial_ESD_Internal Class Reference	76
10.2.1	Detailed Description	76
10.2.2	Member Data Documentation	77
10.2.2.1	m_cmsg	77
10.2.2.2	m_cmsg_next	77
10.2.2.3	ntcan_handle	77
10.2.2.4	rc	77
10.2.2.5	timeout_ms	77
10.3	SDH::cCANSerial_ESDException Class Reference	77
10.3.1	Detailed Description	80
10.3.2	Constructor & Destructor Documentation	80
10.3.2.1	cCANSerial_ESDException	80
10.4	SDH::cCANSerial_PEAK Class Reference	80
10.4.1	Detailed Description	84
10.4.2	Constructor & Destructor Documentation	84
10.4.2.1	cCANSerial_PEAK	84
10.4.2.2	cCANSerial_PEAK	84
10.4.2.3	~cCANSerial_PEAK	84

10.4.3	Member Function Documentation	85
10.4.3.1	BaudrateToBaudrateCode	85
10.4.3.2	Close	85
10.4.3.3	GetErrorMessage	85
10.4.3.4	GetErrorNumber	85
10.4.3.5	GetHandle	85
10.4.3.6	IsOpen	86
10.4.3.7	Open	86
10.4.3.8	Read	86
10.4.3.9	SetTimeout	86
10.4.3.10	write	86
10.4.4	Member Data Documentation	87
10.4.4.1	baudrate	87
10.4.4.2	id_read	87
10.4.4.3	id_write	87
10.4.4.4	m_device	87
10.5	SDH::cCANSerial_PEAK_Internal Class Reference	87
10.5.1	Detailed Description	87
10.5.2	Member Data Documentation	88
10.5.2.1	m_msg	88
10.5.2.2	m_msg_next	88
10.5.2.3	peak_handle	88
10.5.2.4	rc	88
10.6	SDH::cCANSerial_PEAKException Class Reference	88
10.6.1	Detailed Description	91
10.6.2	Constructor & Destructor Documentation	91
10.6.2.1	cCANSerial_PEAKException	91
10.7	SDH::cCRC Class Reference	91
10.7.1	Detailed Description	93
10.7.2	Constructor & Destructor Documentation	93
10.7.2.1	cCRC	93
10.7.3	Member Function Documentation	94
10.7.3.1	AddByte	94
10.7.3.2	AddBytes	94
10.7.3.3	GetCRC	94
10.7.3.4	GetCRC_HB	94
10.7.3.5	GetCRC_LB	94
10.7.3.6	Reset	94
10.7.4	Member Data Documentation	94
10.7.4.1	crc_table	94
10.7.4.2	current_crc	94
10.7.4.3	initial_value	94
10.8	SDH::cCRC_DSACON32m Class Reference	95
10.8.1	Detailed Description	97
10.8.2	Constructor & Destructor Documentation	97
10.8.2.1	cCRC_DSACON32m	97
10.8.3	Member Data Documentation	97
10.8.3.1	crc_table_dsacon32m	97
10.9	SDH::cCRC_SDH Class Reference	98
10.9.1	Detailed Description	101

10.9.2	Constructor & Destructor Documentation	101
10.9.2.1	cCRC_SDH	101
10.10	SDH::cDBG Class Reference	101
10.10.1	Detailed Description	101
10.10.2	Constructor & Destructor Documentation	102
10.10.2.1	cDBG	102
10.10.2.2	~cDBG	102
10.10.3	Member Function Documentation	102
10.10.3.1	GetFlag	102
10.10.3.2	PDM	103
10.10.3.3	SetColor	103
10.10.3.4	SetFlag	103
10.10.3.5	SetOutput	103
10.10.4	Member Data Documentation	103
10.10.4.1	debug_color	103
10.10.4.2	debug_flag	103
10.10.4.3	mywidth	103
10.10.4.4	normal_color	103
10.10.4.5	output	103
10.11	SDH::cDSA Class Reference	103
10.11.1	Detailed Description	109
10.11.2	Member Typedef Documentation	109
10.11.2.1	tTexel	109
10.11.3	Member Enumeration Documentation	109
10.11.3.1	eDSASensorInfo	109
10.11.4	Constructor & Destructor Documentation	110
10.11.4.1	cDSA	110
10.11.4.2	cDSA	111
10.11.4.3	~cDSA	111
10.11.5	Member Function Documentation	111
10.11.5.1	Close	111
10.11.5.2	ErrorCodeToString	111
10.11.5.3	ErrorCodeToString	112
10.11.5.4	GetAgeOfFrame	112
10.11.5.5	GetContactArea	112
10.11.5.6	GetContactInfo	112
10.11.5.7	GetControllerInfo	112
10.11.5.8	GetFrame	112
10.11.5.9	GetMatrixIndex	112
10.11.5.10	GetMatrixInfo	112
10.11.5.11	GetMatrixSensitivity	113
10.11.5.12	GetMatrixThreshold	113
10.11.5.13	GetSensorInfo	113
10.11.5.14	GetTexel	113
10.11.5.15	Open	114
10.11.5.16	ParseFrame	114
10.11.5.17	QueryControllerInfo	114
10.11.5.18	QueryMatrixInfo	114
10.11.5.19	QueryMatrixInfos	114
10.11.5.20	QuerySensorInfo	114

10.11.5.21ReadControllerInfo	114
10.11.5.22ReadFrame	114
10.11.5.23ReadMatrixInfo	115
10.11.5.24ReadResponse	115
10.11.5.25ReadSensorInfo	115
10.11.5.26SDH__attribute__	116
10.11.5.27SDH__attribute__	116
10.11.5.28SDH__attribute__	116
10.11.5.29SDH__attribute__	116
10.11.5.30SDH__attribute__	116
10.11.5.31SetFramerate	116
10.11.5.32SetFramerateRetries	117
10.11.5.33SetMatrixSensitivity	117
10.11.5.34SetMatrixThreshold	118
10.11.5.35UpdateFrame	118
10.11.5.36WriteCommand	119
10.11.5.37WriteCommandWithPayload	119
10.11.6 Friends And Related Function Documentation	119
10.11.6.1 operator<<	119
10.11.7 Member Data Documentation	119
10.11.7.1 calib_pressure	119
10.11.7.2 calib_voltage	119
10.11.7.3 com	119
10.11.7.4 contact_area_cell_threshold	119
10.11.7.5 contact_force_cell_threshold	119
10.11.7.6 controller_info	119
10.11.7.7 dbg	119
10.11.7.8 do_RLE	120
10.11.7.9 force_factor	120
10.11.7.10frame	120
10.11.7.11lm_read_another	120
10.11.7.12matrix_info	120
10.11.7.13nb_cells	120
10.11.7.14read_timeout_us	120
10.11.7.15SDH__attribute__	120
10.11.7.16sensor_info	120
10.11.7.17start_dsa	121
10.11.7.18start_pc	121
10.11.7.19texel_offset	121
10.12SDH::cDSAException Class Reference	121
10.12.1 Detailed Description	122
10.12.2 Constructor & Destructor Documentation	123
10.12.2.1 cDSAException	123
10.13cDSAUpdater Class Reference	123
10.13.1 Detailed Description	123
10.13.2 Constructor & Destructor Documentation	123
10.13.2.1 cDSAUpdater	123
10.13.3 Member Function Documentation	124
10.13.3.1 interrupt	124
10.13.4 Member Data Documentation	124

10.13.4.1	DEFAULT_ERROR_THRESHOLD	124
10.14	SDH::cHexString Class Reference	124
10.14.1	Detailed Description	125
10.14.2	Constructor & Destructor Documentation	125
10.14.2.1	cHexString	125
10.14.3	Friends And Related Function Documentation	125
10.14.3.1	operator<<	125
10.15	cIsGraspedBase Class Reference	125
10.15.1	Detailed Description	128
10.15.2	Constructor & Destructor Documentation	128
10.15.2.1	cIsGraspedBase	128
10.15.3	Member Function Documentation	128
10.15.3.1	IsGrasped	128
10.15.4	Member Data Documentation	128
10.15.4.1	ts	128
10.16	cIsGraspedByArea Class Reference	128
10.16.1	Detailed Description	131
10.16.2	Constructor & Destructor Documentation	131
10.16.2.1	cIsGraspedByArea	131
10.16.3	Member Function Documentation	131
10.16.3.1	IsGrasped	131
10.16.3.2	SetCondition	131
10.16.3.3	SetCondition	132
10.17	SDH::cMsg Class Reference	132
10.17.1	Detailed Description	133
10.17.2	Member Enumeration Documentation	133
10.17.2.1	"@1	133
10.17.3	Constructor & Destructor Documentation	133
10.17.3.1	cMsg	133
10.17.3.2	cMsg	133
10.17.3.3	cMsg	133
10.17.4	Member Function Documentation	133
10.17.4.1	c_str	133
10.17.5	Member Data Documentation	134
10.17.5.1	msg	134
10.18	SDH::cRS232 Class Reference	134
10.18.1	Detailed Description	138
10.18.2	Constructor & Destructor Documentation	138
10.18.2.1	cRS232	138
10.18.2.2	cRS232	139
10.18.2.3	~cRS232	139
10.18.3	Member Function Documentation	139
10.18.3.1	BaudrateToBaudrateCode	139
10.18.3.2	Close	139
10.18.3.3	Close	139
10.18.3.4	IsOpen	139
10.18.3.5	IsOpen	139
10.18.3.6	Open	140
10.18.3.7	Open	140
10.18.3.8	Read	140

10.18.3.9 Read	140
10.18.3.10readline	141
10.18.3.11SetTimeout	141
10.18.3.12UseCRC16	141
10.18.3.13UseCRC16	141
10.18.3.14write	141
10.18.3.15write	141
10.18.4 Member Data Documentation	142
10.18.4.1 baudrate	142
10.18.4.2 device_format_string	142
10.18.4.3 fd	142
10.18.4.4 io_set_old	142
10.18.4.5 port	142
10.18.4.6 status	142
10.19SDH::cRS232Exception Class Reference	142
10.19.1 Detailed Description	145
10.19.2 Constructor & Destructor Documentation	145
10.19.2.1 cRS232Exception	145
10.19.2.2 cRS232Exception	145
10.20SDH::cSDH Class Reference	145
10.20.1 Detailed Description	156
10.20.2 Member Enumeration Documentation	157
10.20.2.1 eAxisState	157
10.20.2.2 eMotorCurrentMode	157
10.20.3 Constructor & Destructor Documentation	157
10.20.3.1 cSDH	157
10.20.3.2 ~cSDH	159
10.20.4 Member Function Documentation	159
10.20.4.1 _GetFingerXYZ	159
10.20.4.2 CheckFirmwareRelease	159
10.20.4.3 Close	160
10.20.4.4 EmergencyStop	161
10.20.4.5 GetAxisActualAngle	161
10.20.4.6 GetAxisActualAngle	162
10.20.4.7 GetAxisActualState	162
10.20.4.8 GetAxisActualState	163
10.20.4.9 GetAxisActualVelocity	163
10.20.4.10GetAxisActualVelocity	164
10.20.4.11GetAxisEnable	164
10.20.4.12GetAxisEnable	164
10.20.4.13GetAxisLimitAcceleration	165
10.20.4.14GetAxisLimitAcceleration	166
10.20.4.15GetAxisLimitVelocity	166
10.20.4.16GetAxisLimitVelocity	166
10.20.4.17GetAxisMaxAcceleration	167
10.20.4.18GetAxisMaxAcceleration	168
10.20.4.19GetAxisMaxAngle	168
10.20.4.20GetAxisMaxAngle	168
10.20.4.21GetAxisMaxVelocity	169
10.20.4.22GetAxisMaxVelocity	169

10.20.4.23GetAxisMinAngle	170
10.20.4.24GetAxisMinAngle	171
10.20.4.25GetAxisMotorCurrent	172
10.20.4.26GetAxisMotorCurrent	173
10.20.4.27GetAxisReferenceVelocity	173
10.20.4.28GetAxisReferenceVelocity	174
10.20.4.29GetAxisTargetAcceleration	174
10.20.4.30GetAxisTargetAcceleration	175
10.20.4.31GetAxisTargetAngle	175
10.20.4.32GetAxisTargetAngle	175
10.20.4.33GetAxisTargetVelocity	176
10.20.4.34GetAxisTargetVelocity	177
10.20.4.35GetAxisValueVector	177
10.20.4.36GetController	178
10.20.4.37GetFingerActualAngle	178
10.20.4.38GetFingerActualAngle	179
10.20.4.39GetFingerAxisIndex	179
10.20.4.40GetFingerEnable	179
10.20.4.41GetFingerEnable	180
10.20.4.42GetFingerMaxAngle	180
10.20.4.43GetFingerMaxAngle	181
10.20.4.44GetFingerMinAngle	181
10.20.4.45GetFingerMinAngle	181
10.20.4.46GetFingerNumberOfAxes	182
10.20.4.47GetFingerTargetAngle	183
10.20.4.48GetFingerTargetAngle	183
10.20.4.49GetFingerXYZ	183
10.20.4.50GetFingerXYZ	184
10.20.4.51GetFirmwareRelease	185
10.20.4.52GetFirmwareReleaseRecommended	185
10.20.4.53GetGripMaxVelocity	185
10.20.4.54GetInfo	186
10.20.4.55GetLibraryName	186
10.20.4.56GetLibraryRelease	187
10.20.4.57GetMotorCurrentModeFunction	187
10.20.4.58GetTemperature	187
10.20.4.59GetTemperature	188
10.20.4.60GetVelocityProfile	188
10.20.4.61GripHand	188
10.20.4.62IsOpen	190
10.20.4.63IsVirtualAxis	190
10.20.4.64MoveAxis	190
10.20.4.65MoveAxis	192
10.20.4.66MoveFinger	192
10.20.4.67MoveFinger	193
10.20.4.68MoveHand	194
10.20.4.69OpenCAN_ESD	195
10.20.4.70OpenCAN_ESD	195
10.20.4.71OpenCAN_PEAK	196
10.20.4.72OpenCAN_PEAK	197

10.20.4.73	OpenRS232	197
10.20.4.74	OpenTCP	198
10.20.4.75	SetAxisEnable	199
10.20.4.76	SetAxisEnable	199
10.20.4.77	SetAxisEnable	200
10.20.4.78	SetAxisEnable	200
10.20.4.79	SetAxisMotorCurrent	200
10.20.4.80	SetAxisMotorCurrent	200
10.20.4.81	SetAxisTargetAcceleration	202
10.20.4.82	SetAxisTargetAcceleration	202
10.20.4.83	SetAxisTargetAngle	203
10.20.4.84	SetAxisTargetAngle	204
10.20.4.85	SetAxisTargetGetAxisActualAngle	205
10.20.4.86	SetAxisTargetGetAxisActualVelocity	206
10.20.4.87	SetAxisTargetVelocity	208
10.20.4.88	SetAxisTargetVelocity	208
10.20.4.89	SetAxisValueVector	209
10.20.4.90	SetController	210
10.20.4.91	SetDebugOutput	211
10.20.4.92	SetFingerEnable	212
10.20.4.93	SetFingerEnable	213
10.20.4.94	SetFingerEnable	213
10.20.4.95	SetFingerEnable	213
10.20.4.96	SetFingerTargetAngle	213
10.20.4.97	SetFingerTargetAngle	214
10.20.4.98	SetVelocityProfile	214
10.20.4.99	Stop	215
10.20.4.100	IndexVector	215
10.20.4.101	UseDegrees	216
10.20.4.102	UseRadians	216
10.20.4.103	WaitAxis	216
10.20.4.104	WaitAxis	216
10.20.5	Member Data Documentation	219
10.20.5.1	all_axes	219
10.20.5.2	all_fingers	219
10.20.5.3	all_real_axes	219
10.20.5.4	all_temperature_sensors	219
10.20.5.5	com	219
10.20.5.6	comm_interface	219
10.20.5.7	d	219
10.20.5.8	f_max_acceleration_v	219
10.20.5.9	f_max_angle_v	219
10.20.5.10	f_max_motor_current_v	220
10.20.5.11	f_max_velocity_v	220
10.20.5.12	f_min_acceleration_v	220
10.20.5.13	f_min_angle_v	220
10.20.5.14	f_min_motor_current_v	220
10.20.5.15	f_min_velocity_v	220
10.20.5.16	f_ones_v	220
10.20.5.17	f_zeros_v	220

10.20.5.18	finger_axis_index	221
10.20.5.19	finger_number_of_axes	221
10.20.5.20	grip_max_velocity	221
10.20.5.21	lh	221
10.20.5.22	1	221
10.20.5.23	2	221
10.20.5.24	nb_all_axes	221
10.20.5.25	NUMBER_OF_AXES_PER_FINGER	221
10.20.5.26	NUMBER_OF_VIRTUAL_AXES	221
10.20.5.27	offset	221
10.20.5.28	ones_v	221
10.20.5.29	uc_angle	222
10.20.5.30	uc_angle_degrees	222
10.20.5.31	uc_angle_radians	222
10.20.5.32	uc_angular_acceleration	222
10.20.5.33	uc_angular_acceleration_degrees_per_second_squared	222
10.20.5.34	uc_angular_acceleration_radians_per_second_squared	222
10.20.5.35	uc_angular_velocity	222
10.20.5.36	uc_angular_velocity_degrees_per_second	222
10.20.5.37	uc_angular_velocity_radians_per_second	222
10.20.5.38	uc_motor_current	223
10.20.5.39	uc_motor_current_ampere	223
10.20.5.40	uc_motor_current_milliampere	223
10.20.5.41	uc_position	223
10.20.5.42	uc_position_meter	223
10.20.5.43	uc_position_millimeter	223
10.20.5.44	uc_temperature	223
10.20.5.45	uc_temperature_celsius	223
10.20.5.46	uc_temperature_fahrenheit	223
10.20.5.47	uc_time	223
10.20.5.48	uc_time_milliseconds	224
10.20.5.49	uc_time_seconds	224
10.20.5.50	zeros_v	224
10.21	SDH::cSDHBase Class Reference	224
10.21.1	Detailed Description	230
10.21.2	Member Enumeration Documentation	230
10.21.2.1	"@0	230
10.21.2.2	eControllerType	230
10.21.2.3	eErrorCode	231
10.21.2.4	eGraspId	232
10.21.2.5	eVelocityProfile	232
10.21.3	Constructor & Destructor Documentation	233
10.21.3.1	cSDHBase	233
10.21.3.2	~cSDHBase	233
10.21.4	Member Function Documentation	233
10.21.4.1	CheckIndex	233
10.21.4.2	CheckRange	233
10.21.4.3	CheckRange	233
10.21.4.4	GetEps	233
10.21.4.5	GetEpsVector	233

10.21.4.6	GetFirmwareState	234
10.21.4.7	GetNumberOfAxes	234
10.21.4.8	GetNumberOfFingers	234
10.21.4.9	GetNumberOfTemperatureSensors	234
10.21.4.10	GetStringFromControllerType	234
10.21.4.11	GetStringFromErrorCode	234
10.21.4.12	GetStringFromGraspId	234
10.21.4.13	IsOpen	234
10.21.4.14	SetDebugOutput	234
10.21.5	Member Data Documentation	235
10.21.5.1	all_axes_used	235
10.21.5.2	cdbg	235
10.21.5.3	controller_type_name	235
10.21.5.4	debug_level	235
10.21.5.5	eps	235
10.21.5.6	eps_v	235
10.21.5.7	firmware_error_codes	235
10.21.5.8	firmware_state	236
10.21.5.9	grasp_id_name	237
10.21.5.10	max_angle_v	237
10.21.5.11	min_angle_v	237
10.21.5.12	NUMBER_OF_AXES	237
10.21.5.13	NUMBER_OF_FINGERS	237
10.21.5.14	NUMBER_OF_TEMPERATURE_SENSORS	237
10.22	SDH::cSDHErrorCommunication Class Reference	238
10.22.1	Detailed Description	240
10.22.2	Constructor & Destructor Documentation	240
10.22.2.1	cSDHErrorCommunication	240
10.22.2.2	cSDHErrorCommunication	240
10.23	SDH::cSDHErrorInvalidParameter Class Reference	240
10.23.1	Detailed Description	241
10.23.2	Constructor & Destructor Documentation	242
10.23.2.1	cSDHErrorInvalidParameter	242
10.24	SDH::cSDHLibraryException Class Reference	242
10.24.1	Detailed Description	243
10.24.2	Constructor & Destructor Documentation	244
10.24.2.1	cSDHLibraryException	244
10.24.3	Member Function Documentation	245
10.24.3.1	what	245
10.24.4	Member Data Documentation	245
10.24.4.1	msg	245
10.25	cSDHOptions Class Reference	245
10.25.1	Detailed Description	246
10.25.2	Constructor & Destructor Documentation	247
10.25.2.1	cSDHOptions	247
10.25.2.2	~cSDHOptions	247
10.25.3	Member Function Documentation	247
10.25.3.1	OpenCommunication	247
10.25.3.2	Parse	248
10.25.4	Member Data Documentation	250

10.25.4.1	can_baudrate	250
10.25.4.2	controllerinfo	250
10.25.4.3	debug_level	250
10.25.4.4	debuglog	250
10.25.4.5	do_RLE	250
10.25.4.6	dsa_rs_device	250
10.25.4.7	dsa_tcp_port	250
10.25.4.8	dsa_use_tcp	250
10.25.4.9	dsaport	250
10.25.4.10	framerate	250
10.25.4.11	fullframe	250
10.25.4.12	id_read	250
10.25.4.13	id_write	250
10.25.4.14	matrixinfo	250
10.25.4.15	MAX_DEV_LENGTH	250
10.25.4.16	net	250
10.25.4.17	period	250
10.25.4.18	persistent	250
10.25.4.19	reset_to_default	250
10.25.4.20	rs232_baudrate	250
10.25.4.21	sdh_canpeak_device	250
10.25.4.22	sdh_rs_device	250
10.25.4.23	sdhport	250
10.25.4.24	sensitivity	250
10.25.4.25	sensorinfo	250
10.25.4.26	showdsasettings	250
10.25.4.27	tcp_adr	250
10.25.4.28	tcp_port	250
10.25.4.29	threshold	250
10.25.4.30	timeout	250
10.25.4.31	usage	250
10.25.4.32	use_can_esd	250
10.25.4.33	use_can_peak	250
10.25.4.34	use_fahrenheit	250
10.25.4.35	use_radians	250
10.25.4.36	use_tcp	250
10.26	SDH::cSDHSerial Class Reference	251
10.26.1	Detailed Description	256
10.26.2	Constructor & Destructor Documentation	256
10.26.2.1	cSDHSerial	256
10.26.2.2	~cSDHSerial	256
10.26.3	Member Function Documentation	256
10.26.3.1	a	256
10.26.3.2	alim	257
10.26.3.3	AxisCommand	257
10.26.3.4	BinaryAxisCommand	257
10.26.3.5	BinarySync	258
10.26.3.6	Close	258
10.26.3.7	con	258
10.26.3.8	debug	258

10.26.3.9 demo	258
10.26.3.10 ExtractFirmwareState	258
10.26.3.11 get_duration	258
10.26.3.12 GetDuration	258
10.26.3.13 grip	258
10.26.3.14 id	259
10.26.3.15 grip	259
10.26.3.16 hold	259
10.26.3.17 lim	260
10.26.3.18 isOpen	260
10.26.3.19 kv	260
10.26.3.20 m	261
10.26.3.21 numaxis	261
10.26.3.22 Open	261
10.26.3.23 p	261
10.26.3.24 pid	262
10.26.3.25 pos	262
10.26.3.26 pos_save	263
10.26.3.27 power	263
10.26.3.28 property	263
10.26.3.29 ref	264
10.26.3.30 rvel	264
10.26.3.31 selgrip	265
10.26.3.32 Send	265
10.26.3.33 sn	265
10.26.3.34 soc	265
10.26.3.35 soc_date	265
10.26.3.36 state	266
10.26.3.37 stop	266
10.26.3.38 Sync	266
10.26.3.39 SyncUnknown	266
10.26.3.40 temp	266
10.26.3.41 temp_electronics	266
10.26.3.42 terminal	267
10.26.3.43 tap	267
10.26.3.44 tvav	267
10.26.3.45 user_errors	268
10.26.3.46 v	268
10.26.3.47 vel	268
10.26.3.48 ver	269
10.26.3.49 ver_date	269
10.26.3.50 vlim	269
10.26.3.51 vp	269
10.26.4 Friends And Related Function Documentation	270
10.26.4.1 sSDHBinaryRequest	270
10.26.4.2 sSDHBinaryResponse	270
10.26.5 Member Data Documentation	270
10.26.5.1 com	270
10.26.5.2 EOL	270
10.26.5.3 m_severtime	270

10.26.5.4	nb_lines_to_ignore	270
10.26.5.5	reply	270
10.27	SDH::cSerialBase Class Reference	270
10.27.1	Detailed Description	274
10.27.2	Member Typedef Documentation	274
10.27.2.1	tErrorCode	274
10.27.3	Constructor & Destructor Documentation	274
10.27.3.1	cSerialBase	274
10.27.3.2	~cSerialBase	275
10.27.4	Member Function Documentation	275
10.27.4.1	Close	275
10.27.4.2	GetErrorMessage	275
10.27.4.3	GetErrorNumber	275
10.27.4.4	GetLastErrorMessage	275
10.27.4.5	GetTimeout	275
10.27.4.6	IsOpen	276
10.27.4.7	Open	276
10.27.4.8	Read	276
10.27.4.9	readline	276
10.27.4.10	SetTimeout	277
10.27.4.11	UseCRC16	277
10.27.4.12	write	277
10.27.5	Member Data Documentation	277
10.27.5.1	dbg	277
10.27.5.2	timeout	277
10.27.5.3	ungetch	277
10.27.5.4	ungetch_valid	278
10.28	SDH::cSerialBaseException Class Reference	278
10.28.1	Detailed Description	280
10.28.2	Constructor & Destructor Documentation	280
10.28.2.1	cSerialBaseException	280
10.28.2.2	~cSerialBaseException	280
10.29	SDH::cSerialBase::cSetTimeoutTemporarily Class Reference	280
10.29.1	Detailed Description	282
10.29.2	Constructor & Destructor Documentation	282
10.29.2.1	cSetTimeoutTemporarily	282
10.29.2.2	~cSetTimeoutTemporarily	282
10.30	SDH::cSetValueTemporarily< T > Class Template Reference	282
10.30.1	Detailed Description	283
10.30.2	Constructor & Destructor Documentation	283
10.30.2.1	cSetValueTemporarily	283
10.30.2.2	~cSetValueTemporarily	283
10.31	SDH::cSimpleStringList Class Reference	283
10.31.1	Detailed Description	284
10.31.2	Member Enumeration Documentation	284
10.31.2.1	"@10	284
10.31.3	Constructor & Destructor Documentation	285
10.31.3.1	cSimpleStringList	285
10.31.4	Member Function Documentation	285
10.31.4.1	CurrentLine	285

10.31.4.2 Length	285
10.31.4.3 NextLine	285
10.31.4.4 operator[]	285
10.31.4.5 operator[]	285
10.31.4.6 Reset	285
10.31.5 Member Data Documentation	285
10.31.5.1 current_line	285
10.31.5.2 line	286
10.32SDH::cSimpleTime Class Reference	286
10.32.1 Detailed Description	287
10.32.2 Constructor & Destructor Documentation	287
10.32.2.1 cSimpleTime	287
10.32.3 Member Function Documentation	287
10.32.3.1 Elapsed	287
10.32.3.2 Elapsed	287
10.32.3.3 Elapsed_us	287
10.32.3.4 Elapsed_us	287
10.32.3.5 StoreNow	287
10.32.3.6 Timeval	287
10.32.4 Member Data Documentation	287
10.32.4.1 a_time	287
10.33SDH::cSimpleVector Class Reference	288
10.33.1 Detailed Description	289
10.33.2 Member Enumeration Documentation	289
10.33.2.1 "@11	289
10.33.3 Constructor & Destructor Documentation	289
10.33.3.1 cSimpleVector	289
10.33.3.2 cSimpleVector	289
10.33.3.3 cSimpleVector	290
10.33.3.4 cSimpleVector	290
10.33.4 Member Function Documentation	290
10.33.4.1 FromString	290
10.33.4.2 operator[]	290
10.33.4.3 Valid	290
10.33.4.4 x	290
10.33.4.5 y	290
10.33.4.6 z	290
10.33.5 Member Data Documentation	290
10.33.5.1 valid	290
10.33.5.2 value	291
10.34SDH::cSimpleVectorException Class Reference	291
10.34.1 Detailed Description	292
10.34.2 Constructor & Destructor Documentation	293
10.34.2.1 cSimpleVectorException	293
10.35SDH::cTCPSerial Class Reference	293
10.35.1 Detailed Description	296
10.35.2 Constructor & Destructor Documentation	297
10.35.2.1 cTCPSerial	297
10.35.3 Member Function Documentation	297
10.35.3.1 Close	297

10.35.3.2	GetErrorNumber	297
10.35.3.3	IsOpen	297
10.35.3.4	Open	297
10.35.3.5	Read	298
10.35.3.6	SetTimeout	298
10.35.3.7	write	298
10.35.4	Member Data Documentation	298
10.35.4.1	fd	298
10.35.4.2	INVALID_SOCKET	299
10.35.4.3	tcp_adr	299
10.35.4.4	tcp_port	299
10.35.4.5	TIMEOUT_RETURN_IMMEDIATELY_S	299
10.35.4.6	TIMEOUT_RETURN_IMMEDIATELY_US	299
10.35.4.7	TIMEOUT_WAIT_FOR_EVER_S	299
10.35.4.8	TIMEOUT_WAIT_FOR_EVER_US	299
10.36	SDH::cTCPSerialException Class Reference	299
10.36.1	Detailed Description	302
10.36.2	Constructor & Destructor Documentation	302
10.36.2.1	cTCPSerialException	302
10.37	SDH::cUnitConverter Class Reference	302
10.37.1	Detailed Description	303
10.37.2	Constructor & Destructor Documentation	303
10.37.2.1	cUnitConverter	303
10.37.3	Member Function Documentation	304
10.37.3.1	GetDecimalPlaces	304
10.37.3.2	GetFactor	304
10.37.3.3	GetKind	304
10.37.3.4	GetName	304
10.37.3.5	GetOffset	304
10.37.3.6	GetSymbol	304
10.37.3.7	ToExternal	305
10.37.3.8	ToExternal	305
10.37.3.9	ToExternal	305
10.37.3.10	ToInternal	305
10.37.3.11	ToInternal	305
10.37.3.12	ToInternal	305
10.37.4	Member Data Documentation	305
10.37.4.1	decimal_places	305
10.37.4.2	factor	306
10.37.4.3	kind	306
10.37.4.4	name	306
10.37.4.5	offset	306
10.37.4.6	symbol	306
10.38	option Struct Reference	306
10.38.1	Member Data Documentation	307
10.38.1.1	flag	307
10.38.1.2	has_arg	307
10.38.1.3	name	307
10.38.1.4	val	307
10.39	SDH::cDSA::sContactInfo Struct Reference	307

10.39.1 Detailed Description	307
10.39.2 Member Data Documentation	307
10.39.2.1 area	307
10.39.2.2 cog_x	308
10.39.2.3 cog_y	308
10.39.2.4 force	308
10.40SDH::cDSA::sControllerInfo Struct Reference	308
10.40.1 Detailed Description	308
10.40.2 Member Data Documentation	309
10.40.2.1 active_interface	309
10.40.2.2 can_baudrate	309
10.40.2.3 can_id	309
10.40.2.4 error_code	309
10.40.2.5 feature_flags	309
10.40.2.6 hw_version	309
10.40.2.7 senscon_type	309
10.40.2.8 serial_no	309
10.40.2.9 status_flags	309
10.40.2.10sw_version	309
10.41SDH::cDSA::sMatrixInfo Struct Reference	309
10.41.1 Detailed Description	310
10.41.2 Member Data Documentation	310
10.41.2.1 cells_x	310
10.41.2.2 cells_y	310
10.41.2.3 error_code	310
10.41.2.4 feature_flags	310
10.41.2.5 fullscale	310
10.41.2.6 hw_revision	310
10.41.2.7 matrix_center_x	310
10.41.2.8 matrix_center_y	310
10.41.2.9 matrix_center_z	310
10.41.2.10matrix_theta_x	310
10.41.2.11matrix_theta_y	310
10.41.2.12matrix_theta_z	310
10.41.2.13reserved	310
10.41.2.14texel_height	310
10.41.2.15texel_width	310
10.41.2.16uid	310
10.42sRecordedData Struct Reference	311
10.42.1 Detailed Description	311
10.42.2 Constructor & Destructor Documentation	311
10.42.2.1 sRecordedData	311
10.42.3 Member Data Documentation	311
10.42.3.1 aaa	311
10.42.3.2 aav	311
10.42.3.3 atv	311
10.42.3.4 t	311
10.43SDH::cDSA::sResponse Struct Reference	311
10.43.1 Detailed Description	312
10.43.2 Constructor & Destructor Documentation	312

10.43.2.1 sResponse	312
10.43.3 Member Data Documentation	312
10.43.3.1 max_payload_size	312
10.43.3.2 packet_id	312
10.43.3.3 payload	312
10.43.3.4 size	312
10.44SDH::sSDHBinaryRequest Struct Reference	312
10.44.1 Detailed Description	313
10.44.2 Constructor & Destructor Documentation	313
10.44.2.1 sSDHBinaryRequest	313
10.44.3 Member Function Documentation	313
10.44.3.1 CRC16	313
10.44.3.2 GetNbBytesToSend	313
10.44.4 Member Data Documentation	314
10.44.4.1 __attribute__	314
10.44.4.2 cmd_code	314
10.44.4.3 nb_data_bytes	314
10.44.4.4 nb_valid_parameters	314
10.44.4.5 parameter	314
10.44.4.6 parameter_bytes	314
10.45SDH::sSDHBinaryResponse Struct Reference	314
10.45.1 Detailed Description	315
10.45.2 Member Function Documentation	315
10.45.2.1 CheckCRC16	315
10.45.2.2 CRC16	315
10.45.3 Member Data Documentation	315
10.45.3.1 "@7	315
10.45.3.2 cmd_code	315
10.45.3.3 nb_data_bytes	315
10.45.3.4 nb_valid_parameters	315
10.45.3.5 parameter	315
10.45.3.6 parameter_bytes	315
10.45.3.7 status_code	315
10.46SDH::cDSA::sSensitivityInfo Struct Reference	315
10.46.1 Detailed Description	316
10.46.2 Member Data Documentation	316
10.46.2.1 adj_flags	316
10.46.2.2 cur_sens	316
10.46.2.3 error_code	316
10.46.2.4 fact_sens	316
10.47SDH::cDSA::sSensorInfo Struct Reference	317
10.47.1 Detailed Description	317
10.47.2 Member Data Documentation	317
10.47.2.1 error_code	317
10.47.2.2 feature_flags	317
10.47.2.3 generated_by	317
10.47.2.4 hw_revision	317
10.47.2.5 nb_matrices	317
10.47.2.6 serial_no	317
10.48SDH::cDSA::sTactileSensorFrame Struct Reference	317

10.48.1 Detailed Description	318
10.48.2 Constructor & Destructor Documentation	318
10.48.2.1 sTactileSensorFrame	318
10.48.3 Member Data Documentation	318
10.48.3.1 flags	318
10.48.3.2 texel	319
10.48.3.3 timestamp	319
11 File Documentation	321
11.1 architecture.dox File Reference	321
11.1.1 Detailed Description	321
11.1.2 Overview	321
11.2 connectors.dox File Reference	324
11.2.1 Detailed Description	324
11.2.2 General project information	324
11.2.3 Purpose	324
11.2.4 Copyright	328
11.3 demo/cancat.cpp File Reference	328
11.3.1 Detailed Description	329
11.3.2 General file information	329
11.3.3 Copyright	329
11.3.4 Function Documentation	329
11.3.4.1 main	329
11.3.5 Variable Documentation	329
11.3.5.1 __author__	329
11.3.5.2 __copyright__	329
11.3.5.3 __help__	329
11.3.5.4 __url__	329
11.3.5.5 __version__	329
11.4 demo/demo-benchmark.cpp File Reference	329
11.4.1 Detailed Description	331
11.4.2 General file information	331
11.4.3 Copyright	331
11.4.4 Define Documentation	331
11.4.4.1 _USE_MATH_DEFINES	331
11.4.4.2 DEMO_BENCHMARK_USE_COMBINED_SET_- GET	331
11.4.5 Function Documentation	331
11.4.5.1 GotoPose	331
11.4.5.2 main	332
11.4.6 Variable Documentation	332
11.4.6.1 __author__	332
11.4.6.2 __copyright__	332
11.4.6.3 __help__	332
11.4.6.4 __url__	332
11.4.6.5 __version__	332
11.4.6.6 cdbg	332
11.4.6.7 usage	332
11.5 demo/demo-contact-grasping.cpp File Reference	332
11.5.1 Detailed Description	333

11.5.2	General file information	333
11.5.3	Copyright	333
11.5.4	Function Documentation	334
11.5.4.1	AxisAnglesToFingerAngles	334
11.5.4.2	cdbg	334
11.5.4.3	GotoStartPose	334
11.5.4.4	main	334
11.5.5	Variable Documentation	334
11.5.5.1	__author__	334
11.5.5.2	__copyright__	334
11.5.5.3	__help__	334
11.5.5.4	__url__	334
11.5.5.5	__version__	334
11.5.5.6	usage	334
11.6	demo/demo-dsa-simple.cpp File Reference	334
11.6.1	Detailed Description	335
11.6.2	General file information	335
11.6.3	Copyright	335
11.6.4	Function Documentation	335
11.6.4.1	main	335
11.6.5	Variable Documentation	336
11.6.5.1	__author__	336
11.6.5.2	__copyright__	336
11.6.5.3	__help__	336
11.6.5.4	__url__	336
11.6.5.5	__version__	336
11.6.5.6	usage	336
11.7	demo/demo-dsa.cpp File Reference	336
11.7.1	Detailed Description	337
11.7.2	General file information	337
11.7.3	Copyright	337
11.7.4	Function Documentation	338
11.7.4.1	main	338
11.7.5	Variable Documentation	338
11.7.5.1	__author__	338
11.7.5.2	__copyright__	338
11.7.5.3	__help__	338
11.7.5.4	__url__	338
11.7.5.5	__version__	338
11.7.5.6	usage	338
11.8	demo/demo-GetAxisActualAngle.cpp File Reference	338
11.8.1	Detailed Description	339
11.8.2	General file information	339
11.8.3	Copyright	340
11.8.4	Function Documentation	340
11.8.4.1	main	340
11.8.5	Variable Documentation	340
11.8.5.1	__author__	340
11.8.5.2	__copyright__	340
11.8.5.3	__help__	340

11.8.5.4	<code>__url__</code>	340
11.8.5.5	<code>__version__</code>	340
11.8.5.6	<code>usage</code>	340
11.9	<code>demo/demo-GetFingerXYZ.cpp</code> File Reference	340
11.9.1	Detailed Description	341
11.9.2	General file information	341
11.9.3	Copyright	342
11.9.4	Function Documentation	342
11.9.4.1	<code>main</code>	342
11.9.5	Variable Documentation	342
11.9.5.1	<code>__author__</code>	342
11.9.5.2	<code>__copyright__</code>	342
11.9.5.3	<code>__help__</code>	342
11.9.5.4	<code>__url__</code>	342
11.9.5.5	<code>__version__</code>	342
11.9.5.6	<code>usage</code>	342
11.10	<code>demo/demo-griphand.cpp</code> File Reference	342
11.10.1	Detailed Description	343
11.10.2	General file information	343
11.10.3	Copyright	344
11.10.4	Function Documentation	344
11.10.4.1	<code>main</code>	344
11.10.5	Variable Documentation	344
11.10.5.1	<code>__author__</code>	344
11.10.5.2	<code>__copyright__</code>	344
11.10.5.3	<code>__help__</code>	344
11.10.5.4	<code>__url__</code>	344
11.10.5.5	<code>__version__</code>	344
11.10.5.6	<code>usage</code>	344
11.11	<code>demo/demo-mimic.cpp</code> File Reference	344
11.11.1	Detailed Description	345
11.11.2	General file information	345
11.11.3	Copyright	346
11.11.4	Function Documentation	346
11.11.4.1	<code>GetHand</code>	346
11.11.4.2	<code>main</code>	346
11.11.5	Variable Documentation	346
11.11.5.1	<code>__author__</code>	346
11.11.5.2	<code>__copyright__</code>	346
11.11.5.3	<code>__help__</code>	346
11.11.5.4	<code>__url__</code>	346
11.11.5.5	<code>__version__</code>	346
11.12	<code>demo/demo-radians.cpp</code> File Reference	346
11.12.1	Detailed Description	347
11.12.2	General file information	347
11.12.3	Copyright	348
11.12.4	Function Documentation	348
11.12.4.1	<code>main</code>	348
11.12.5	Variable Documentation	348
11.12.5.1	<code>__author__</code>	348

11.12.5.2	<code>__copyright__</code>	348
11.12.5.3	<code>__help__</code>	348
11.12.5.4	<code>__url__</code>	348
11.12.5.5	<code>__version__</code>	348
11.12.5.6	<code>usage</code>	348
11.13	<code>demo/demo-simple-withtiming.cpp</code> File Reference	348
11.13.1	Detailed Description	349
11.13.2	General file information	349
11.13.3	Copyright	350
11.13.4	Function Documentation	350
11.13.4.1	<code>main</code>	350
11.13.5	Variable Documentation	350
11.13.5.1	<code>__author__</code>	350
11.13.5.2	<code>__copyright__</code>	350
11.13.5.3	<code>__help__</code>	350
11.13.5.4	<code>__url__</code>	350
11.13.5.5	<code>__version__</code>	350
11.13.5.6	<code>usage</code>	350
11.14	<code>demo/demo-simple.cpp</code> File Reference	350
11.14.1	Detailed Description	351
11.14.2	General file information	351
11.14.3	Copyright	352
11.14.4	Function Documentation	352
11.14.4.1	<code>main</code>	352
11.14.5	Variable Documentation	352
11.14.5.1	<code>__author__</code>	352
11.14.5.2	<code>__copyright__</code>	352
11.14.5.3	<code>__help__</code>	352
11.14.5.4	<code>__url__</code>	352
11.14.5.5	<code>__version__</code>	352
11.14.5.6	<code>usage</code>	352
11.15	<code>demo/demo-simple2.cpp</code> File Reference	352
11.15.1	Detailed Description	353
11.15.2	General file information	353
11.15.3	Copyright	354
11.15.4	Function Documentation	354
11.15.4.1	<code>main</code>	354
11.15.5	Variable Documentation	354
11.15.5.1	<code>__author__</code>	354
11.15.5.2	<code>__copyright__</code>	354
11.15.5.3	<code>__help__</code>	354
11.15.5.4	<code>__url__</code>	354
11.15.5.5	<code>__version__</code>	354
11.16	<code>demo/demo-simple3.cpp</code> File Reference	354
11.16.1	Detailed Description	355
11.16.2	General file information	355
11.16.3	Copyright	356
11.16.4	Function Documentation	356
11.16.4.1	<code>main</code>	356
11.16.5	Variable Documentation	356

11.16.5.1	__author__	356
11.16.5.2	__copyright__	356
11.16.5.3	__help__	356
11.16.5.4	__url__	356
11.16.5.5	__version__	356
11.17	demo/demo-temperature.cpp File Reference	356
11.17.1	Detailed Description	357
11.17.2	General file information	357
11.17.3	Copyright	357
11.17.4	Function Documentation	358
11.17.4.1	main	358
11.17.5	Variable Documentation	358
11.17.5.1	__author__	358
11.17.5.2	__copyright__	358
11.17.5.3	__help__	358
11.17.5.4	__url__	358
11.17.5.5	__version__	358
11.18	demo/demo-velocity-acceleration.cpp File Reference	358
11.18.1	Detailed Description	359
11.18.2	General file information	359
11.18.3	Copyright	359
11.18.4	Function Documentation	360
11.18.4.1	main	360
11.18.5	Variable Documentation	360
11.18.5.1	__author__	360
11.18.5.2	__copyright__	360
11.18.5.3	__help__	360
11.18.5.4	__url__	360
11.18.5.5	__version__	360
11.18.5.6	usage	360
11.19	demo/dsaboost.cpp File Reference	360
11.19.1	Detailed Description	361
11.19.2	General file information	361
11.19.3	Copyright	361
11.19.4	Variable Documentation	361
11.19.4.1	cdbg	361
11.20	demo/dsaboost.h File Reference	361
11.20.1	Detailed Description	362
11.20.2	General file information	363
11.20.3	Copyright	363
11.21	demo/sdhoptions.cpp File Reference	363
11.21.1	Detailed Description	365
11.21.2	Copyright	365
11.21.3	Define Documentation	365
11.21.3.1	STRINGIFY	365
11.21.3.2	XSTRINGIFY	365
11.21.4	Variable Documentation	365
11.21.4.1	sdhoptions_long_options	365
11.21.4.2	sdhoptions_short_options	365
11.21.4.3	sdhusage_dsaadjust	365

11.21.4.4	sdhusage_dsacom	366
11.21.4.5	sdhusage_dsaothor	366
11.21.4.6	sdhusage_general	366
11.21.4.7	sdhusage_sdhcom_cancommon	366
11.21.4.8	sdhusage_sdhcom_common	366
11.21.4.9	sdhusage_sdhcom_esdcan	366
11.21.4.10	sdhusage_sdhcom_peakcan	366
11.21.4.11	sdhusage_sdhcom_serial	366
11.21.4.12	sdhusage_sdhcom_tcp	366
11.21.4.13	sdhusage_sdhother	366
11.22	demo/sdhoptions.h File Reference	367
11.22.1	Detailed Description	368
11.22.2	General file information	368
11.22.3	Copyright	368
11.22.4	Define Documentation	368
11.22.4.1	DSA_DEFAULT_TCP_PORT	368
11.22.4.2	SDH_DEFAULT_TCP_ADR	368
11.22.4.3	SDH_DEFAULT_TCP_PORT	368
11.22.4.4	SDHUSAGE_DEFAULT	368
11.23	doc/onlinehelp-demo-benchmark.exe.dox File Reference	370
11.23.1	Detailed Description	370
11.24	doc/onlinehelp-demo-contact-grasping.exe.dox File Reference	370
11.24.1	Detailed Description	370
11.25	doc/onlinehelp-demo-dsa-simple.exe.dox File Reference	370
11.25.1	Detailed Description	370
11.26	doc/onlinehelp-demo-dsa-simple.log.dox File Reference	370
11.27	doc/onlinehelp-demo-dsa.exe.dox File Reference	370
11.27.1	Detailed Description	370
11.28	doc/onlinehelp-demo-GetAxisActualAngle.exe.dox File Reference	370
11.28.1	Detailed Description	370
11.29	doc/onlinehelp-demo-GetFingerXYZ.exe.dox File Reference	370
11.29.1	Detailed Description	370
11.30	doc/onlinehelp-demo-griphand.exe.dox File Reference	370
11.30.1	Detailed Description	370
11.31	doc/onlinehelp-demo-mimic.exe.dox File Reference	370
11.31.1	Detailed Description	370
11.32	doc/onlinehelp-demo-radians.exe.dox File Reference	370
11.32.1	Detailed Description	370
11.33	doc/onlinehelp-demo-ref.exe.dox File Reference	370
11.33.1	Detailed Description	370
11.34	doc/onlinehelp-demo-simple-withtiming.exe.dox File Reference	370
11.34.1	Detailed Description	370
11.35	doc/onlinehelp-demo-simple.exe.dox File Reference	370
11.35.1	Detailed Description	370
11.36	doc/onlinehelp-demo-simple2.exe.dox File Reference	370
11.36.1	Detailed Description	370
11.37	doc/onlinehelp-demo-simple3.exe.dox File Reference	370
11.37.1	Detailed Description	370
11.38	doc/onlinehelp-demo-temperature.exe.dox File Reference	370
11.38.1	Detailed Description	370

11.39doc/onlinehelp-demo-test.exe.dox File Reference	370
11.39.1 Detailed Description	370
11.40doc/onlinehelp-demo-velocity-acceleration.exe.dox File Reference	370
11.40.1 Detailed Description	370
11.41Doxyfile File Reference	370
11.41.1 Detailed Description	371
11.41.2 General file information	371
11.41.3 Links	371
11.41.4 Copyright	371
11.42Makefile File Reference	371
11.42.1 Detailed Description	371
11.42.2 General file information	371
11.42.3 Makefile variables	372
11.42.4 Makefile targets	372
11.42.5 Links	372
11.42.6 Copyright	372
11.43sdh/basisdef.h File Reference	372
11.43.1 Detailed Description	374
11.43.2 General file information	374
11.43.3 Copyright	374
11.43.4 Define Documentation	374
11.43.4.1 SDH_ASSERT_TYPESIZES	374
11.44sdh/canserial-esd.cpp File Reference	375
11.44.1 Detailed Description	376
11.44.2 General file information	376
11.44.3 Copyright	376
11.44.4 Define Documentation	376
11.44.4.1 DBG	376
11.44.4.2 SDH_CANSERIAL_ESD_DEBUG	377
11.44.5 Function Documentation	377
11.44.5.1 ESD_strerror	377
11.45sdh/canserial-esd.h File Reference	377
11.45.1 Detailed Description	378
11.45.2 General file information	379
11.45.3 Copyright	379
11.45.4 Define Documentation	379
11.45.4.1 CAN_ESD_RXQUEUE_SIZE	379
11.45.4.2 CAN_ESD_TXQUEUE_SIZE	379
11.46sdh/canserial-peak.cpp File Reference	379
11.46.1 Detailed Description	381
11.46.2 General file information	381
11.46.3 Define Documentation	381
11.46.3.1 DBG	381
11.46.3.2 M_CMSG_MSG	381
11.46.3.3 SDH_CANSERIAL_PEAK_DEBUG	381
11.46.3.4 USE_HANDLE	382
11.46.3.5 USE_HANDLES	382
11.46.4 Function Documentation	382
11.46.4.1 PEAK_strerror	382
11.47sdh/canserial-peak.h File Reference	382

11.47.1 Detailed Description	383
11.47.2 General file information	383
11.48sdh/crc.cpp File Reference	384
11.48.1 Detailed Description	384
11.48.2 General file information	384
11.48.3 Copyright	385
11.49sdh/crc.h File Reference	385
11.49.1 Detailed Description	387
11.49.2 General file information	387
11.49.3 Copyright	387
11.50sdh/dbg.h File Reference	387
11.50.1 Detailed Description	388
11.50.2 General file information	389
11.50.3 Copyright	389
11.50.4 Define Documentation	389
11.50.4.1 VAL	389
11.50.4.2 VAR	389
11.51sdh/dsa.cpp File Reference	389
11.51.1 Detailed Description	391
11.51.2 General file information	391
11.51.3 Copyright	391
11.51.4 Define Documentation	391
11.51.4.1 PRINT_MEMBER	391
11.51.4.2 PRINT_MEMBER_HEX	391
11.51.5 Enumeration Type Documentation	391
11.51.5.1 eDSAPacketID	391
11.52sdh/dsa.h File Reference	392
11.52.1 Detailed Description	395
11.52.2 General file information	396
11.52.3 Copyright	396
11.52.4 Define Documentation	396
11.52.4.1 DSA_MAX_PREAMBLE_SEARCH	396
11.52.5 Function Documentation	396
11.52.5.1 sResponse	396
11.52.6 Variable Documentation	396
11.52.6.1 active_interface	396
11.52.6.2 adj_flags	396
11.52.6.3 can_baudrate	397
11.52.6.4 can_id	397
11.52.6.5 cells_x	397
11.52.6.6 cells_y	397
11.52.6.7 cur_sens	397
11.52.6.8 error_code	397
11.52.6.9 fact_sens	397
11.52.6.10feature_flags	399
11.52.6.11fullscale	399
11.52.6.12generated_by	399
11.52.6.13hw_revision	399
11.52.6.14hw_version	399
11.52.6.15matrix_center_x	399

11.52.6.16	matrix_center_y	399
11.52.6.17	matrix_center_z	399
11.52.6.18	matrix_theta_x	399
11.52.6.19	matrix_theta_y	399
11.52.6.20	matrix_theta_z	399
11.52.6.21	lmax_payload_size	399
11.52.6.22	nb_matrices	399
11.52.6.23	packet_id	399
11.52.6.24	payload	399
11.52.6.25	reserved	399
11.52.6.26	senscon_type	399
11.52.6.27	serial_no	399
11.52.6.28	size	399
11.52.6.29	status_flags	399
11.52.6.30	sw_version	399
11.52.6.31	texel_height	399
11.52.6.32	texel_width	399
11.52.6.33	uid	399
11.53	sdh/release.h File Reference	399
11.53.1	Detailed Description	400
11.53.2	General file information	400
11.53.3	Copyright	401
11.53.4	Define Documentation	401
11.53.4.1	FIRMWARE_RELEASE_RECOMMENDED	401
11.53.4.2	PROJECT_COPYRIGHT	401
11.53.4.3	PROJECT_DATE	401
11.53.4.4	PROJECT_NAME	401
11.53.4.5	PROJECT_RELEASE	401
11.54	sdh/rs232-cygwin.cpp File Reference	414
11.54.1	Detailed Description	415
11.54.2	General file information	416
11.54.3	Copyright	416
11.54.4	Define Documentation	416
11.54.4.1	DBG	416
11.54.4.2	SDH_RS232_CYGWIN_DEBUG	416
11.54.5	Function Documentation	416
11.54.5.1	StrDupNew	416
11.55	sdh/rs232-cygwin.h File Reference	416
11.55.1	Detailed Description	418
11.55.2	General file information	418
11.55.3	Copyright	418
11.56	sdh/rs232-vcc.cpp File Reference	418
11.56.1	Detailed Description	419
11.56.2	General file information	419
11.56.3	Copyright	419
11.56.4	Define Documentation	420
11.56.4.1	_CRT_SECURE_NO_WARNINGS	420
11.56.4.2	DBG	420
11.56.4.3	SDH_RS232_VCC_DEBUG	420
11.57	sdh/rs232-vcc.h File Reference	420

11.57.1 Detailed Description	422
11.57.2 General file information	422
11.57.3 Copyright	422
11.57.4 Define Documentation	422
11.57.4.1 SDH_RS232_VCC_ASYNC	422
11.58sdh/sdh.cpp File Reference	422
11.58.1 Detailed Description	423
11.58.2 General file information	423
11.58.3 Copyright	423
11.58.4 Define Documentation	424
11.58.4.1 _USE_MATH_DEFINES	424
11.59sdh/sdh.h File Reference	424
11.59.1 Detailed Description	425
11.59.2 General file information	425
11.59.3 Copyright	425
11.60sdh/sdh_codes.cpp File Reference	426
11.60.1 Detailed Description	426
11.60.2 General file information	426
11.60.3 Copyright	426
11.61sdh/sdh_codes.h File Reference	427
11.61.1 Detailed Description	428
11.61.2 General file information	428
11.61.3 Copyright	429
11.62sdh/sdh_command_codes.h File Reference	429
11.62.1 Detailed Description	431
11.62.2 General file information	431
11.62.3 Copyright	431
11.62.4 Define Documentation	432
11.62.4.1 SDH__attribute__	432
11.62.4.2 SDH_USE_VCC	432
11.62.4.3 USE_CMD_TEST	432
11.62.5 Typedef Documentation	432
11.62.5.1 eCommandCode	432
11.62.6 Enumeration Type Documentation	432
11.62.6.1 eCommandCodeEnum	432
11.62.7 Function Documentation	434
11.62.7.1 __attribute__	434
11.62.8 Variable Documentation	434
11.62.8.1 CMDC_A	434
11.62.8.2 CMDC ALIM	434
11.62.8.3 CMDC_CHANGE_CHANNEL	434
11.62.8.4 CMDC_CHANGE_RS232	434
11.62.8.5 CMDC_CON	434
11.62.8.6 CMDC_DEBUG	434
11.62.8.7 CMDC_DEMO	434
11.62.8.8 CMDC_GET_DURATION	434
11.62.8.9 CMDC_GRIP	434
11.62.8.10CMDC_ID	434
11.62.8.11CMDC_IGRIP	434
11.62.8.12CMDC_IHOLD	434

11.62.8.13	CMD_C_ILIM	434
11.62.8.14	CMD_C_KV	434
11.62.8.15	CMD_C_M	434
11.62.8.16	CMD_C_NUM_AXIS	434
11.62.8.17	CMD_C_P	434
11.62.8.18	CMD_C_P_MAX	434
11.62.8.19	CMD_C_P_MIN	434
11.62.8.20	CMD_C_P_OFFSET	434
11.62.8.21	CMD_C_PID	434
11.62.8.22	CMD_C_POS	434
11.62.8.23	CMD_C_POS_SAVE	434
11.62.8.24	CMD_C_POWER	434
11.62.8.25	CMD_C_REF	434
11.62.8.26	CMD_C_RVEL	434
11.62.8.27	CMD_C_SELGRIP	434
11.62.8.28	CMD_C_SN	434
11.62.8.29	CMD_C_SOC	434
11.62.8.30	CMD_C_SOC_DATE	434
11.62.8.31	CMD_C_STATE	434
11.62.8.32	CMD_C_STOP	434
11.62.8.33	CMD_C_TEMP	434
11.62.8.34	CMD_C_TERMINAL	434
11.62.8.35	CMD_C_TPAP	434
11.62.8.36	CMD_C_TVAV	434
11.62.8.37	CMD_C_USE_FIXED_LENGTH	434
11.62.8.38	CMD_C_USER_ERRORS	434
11.62.8.39	CMD_C_V	434
11.62.8.40	CMD_C_VEL	434
11.62.8.41	CMD_C_VER	434
11.62.8.42	CMD_C_VER_DATE	434
11.62.8.43	CMD_C_VLIM	434
11.62.8.44	CMD_C_VP	434
11.63	sdh/sdh_return_codes.h File Reference	434
11.63.1	Detailed Description	437
11.63.2	General file information	438
11.63.3	Copyright	438
11.63.4	Define Documentation	438
11.63.4.1	SDH__attribute__	438
11.63.4.2	SDH_USE_VCC	438
11.63.5	Typedef Documentation	438
11.63.5.1	eReturnCode	438
11.63.6	Enumeration Type Documentation	438
11.63.6.1	eReturnCodeEnum	438
11.63.7	Function Documentation	439
11.63.7.1	SDH__attribute__	439
11.63.8	Variable Documentation	439
11.63.8.1	RC_ACCESS_DENIED	439
11.63.8.2	RC_ALREADY_OPEN	439
11.63.8.3	RC_ALREADY_RUNNING	439
11.63.8.4	RC_AXIS_DISABLED	439

11.63.8.5 RC_CHECKSUM_ERROR	439
11.63.8.6 RC_CMD_ABORTED	439
11.63.8.7 RC_CMD_FAILED	439
11.63.8.8 RC_CMD_FORMAT_ERROR	439
11.63.8.9 RC_CMD_UNKNOWN	439
11.63.8.10 RC_CRC_ERROR	439
11.63.8.11 RC_DEVICE_NOT_FOUND	439
11.63.8.12 RC_DEVICE_NOT_OPENED	439
11.63.8.13 RC_DIMENSION	439
11.63.8.14 RC_FEATURE_NOT_SUPPORTED	440
11.63.8.15 RC_HOMING_ERROR	440
11.63.8.16 RC_INCONSISTENT_DATA	440
11.63.8.17 RC_INDEX_OUT_OF_BOUNDS	440
11.63.8.18 RC_INSUFFICIENT_RESOURCES	440
11.63.8.19 RC_INTERNAL	440
11.63.8.20 RC_INVALID_HANDLE	440
11.63.8.21 RC_INVALID_PARAMETER	440
11.63.8.22 RC_INVALID_PASSWORD	440
11.63.8.23 RC_IO_ERROR	440
11.63.8.24 RC_MAX_COMMANDLINE_EXCEEDED	440
11.63.8.25 RC_MAX_COMMANDS_EXCEEDED	440
11.63.8.26 RC_NO_COMMAND	440
11.63.8.27 RC_NO_DATAPIPE	441
11.63.8.28 RC_NO_PARAMS_EXPECTED	441
11.63.8.29 RC_NOT_AVAILABLE	441
11.63.8.30 RC_NOT_ENOUGH_PARAMS	441
11.63.8.31 RC_NOT_INITIALIZED	441
11.63.8.32 RC_OK	441
11.63.8.33 RC_OVER_TEMPERATURE	441
11.63.8.34 RC_RANGE_ERROR	441
11.63.8.35 RC_READ_ERROR	441
11.63.8.36 RC_TIMEOUT	441
11.63.8.37 RC_UNKNOWN_ERROR	441
11.63.8.38 RC_WRITE_ERROR	441
11.64 sdh/sdhbase.cpp File Reference	441
11.64.1 Detailed Description	442
11.64.2 General file information	442
11.64.3 Copyright	442
11.65 sdh/sdhbase.h File Reference	443
11.65.1 Detailed Description	444
11.65.2 General file information	444
11.65.3 Copyright	444
11.66 sdh/sdhexception.cpp File Reference	444
11.66.1 Detailed Description	445
11.66.2 General file information	445
11.66.3 Copyright	445
11.67 sdh/sdhexception.h File Reference	446
11.67.1 Detailed Description	447
11.67.2 General file information	447
11.67.3 Copyright	447

11.68sdh/sdhlbrary_settings.h File Reference	447
11.68.1 Detailed Description	448
11.68.2 General file information	448
11.68.3 Copyright	448
11.69sdh/sdhserial.cpp File Reference	449
11.69.1 Detailed Description	450
11.69.2 General file information	451
11.69.3 Copyright	451
11.69.4 Function Documentation	451
11.69.4.1 CheckCRC16	451
11.69.4.2 CRC16	451
11.69.4.3 GetNbBytesToSend	451
11.69.4.4 sSDHBinaryRequest	451
11.69.5 Variable Documentation	452
11.69.5.1 "@5	452
11.69.5.2 "@9	452
11.69.5.3 cmd_code	452
11.69.5.4 nb_data_bytes	452
11.69.5.5 nb_valid_parameters	452
11.69.5.6 parameter	452
11.69.5.7 parameter_bytes	452
11.69.5.8 status_code	452
11.70sdh/sdhserial.h File Reference	452
11.70.1 Detailed Description	454
11.70.2 General file information	454
11.70.3 Copyright	454
11.71sdh/serialbase.cpp File Reference	454
11.71.1 Detailed Description	455
11.71.2 General file information	455
11.71.3 Copyright	455
11.72sdh/serialbase.h File Reference	455
11.72.1 Detailed Description	457
11.72.2 General file information	457
11.72.3 Copyright	457
11.73sdh/simplestringlist.cpp File Reference	457
11.73.1 Detailed Description	458
11.73.2 General file information	458
11.73.3 Copyright	458
11.74sdh/simplestringlist.h File Reference	459
11.74.1 Detailed Description	460
11.74.2 General file information	460
11.74.3 Copyright	460
11.75sdh/simpleteime.h File Reference	460
11.75.1 Detailed Description	462
11.75.2 General file information	462
11.75.3 Copyright	462
11.76sdh/simplevector.cpp File Reference	462
11.76.1 Detailed Description	462
11.76.2 General file information	463
11.76.3 Copyright	463

11.77	sdh/simplevector.h File Reference	463
11.77.1	Detailed Description	464
11.77.2	General file information	464
11.77.3	Copyright	464
11.78	sdh/tcpserial.cpp File Reference	465
11.78.1	Detailed Description	466
11.78.2	General file information	466
11.78.3	Copyright	466
11.78.4	Define Documentation	466
11.78.4.1	DBG	466
11.78.4.2	SDH_TCP_DEBUG	466
11.79	sdh/tcpserial.h File Reference	466
11.79.1	Detailed Description	468
11.79.2	General file information	468
11.79.3	Copyright	468
11.80	sdh/unit_converter.cpp File Reference	468
11.80.1	Detailed Description	469
11.80.2	General file information	469
11.80.3	Copyright	470
11.81	sdh/unit_converter.h File Reference	470
11.81.1	Detailed Description	471
11.81.2	General file information	472
11.81.3	Copyright	472
11.82	sdh/util.cpp File Reference	472
11.82.1	Detailed Description	474
11.82.2	General file information	474
11.82.3	Copyright	474
11.82.4	Define Documentation	474
11.82.4.1	_USE_MATH_DEFINES	474
11.83	sdh/util.h File Reference	474
11.83.1	Detailed Description	476
11.83.2	General file information	476
11.83.3	Copyright	476
11.83.4	Define Documentation	477
11.83.4.1	DEFINE_TO_CASECOMMAND	477
11.83.4.2	DEFINE_TO_CASECOMMAND_MSG	477
11.84	sdhlibrary_cpp.dox File Reference	478
11.85	vcc/getopt.c File Reference	478
11.85.1	Define Documentation	479
11.85.1.1	_	479
11.85.1.2	_NO_PROTO	479
11.85.1.3	GETOPT_INTERFACE_VERSION	479
11.85.1.4	NONOPTION_P	479
11.85.1.5	SWAP_FLAGS	479
11.85.2	Enumeration Type Documentation	479
11.85.2.1	"@12	479
11.85.3	Function Documentation	481
11.85.3.1	_getopt_initialize	481
11.85.3.2	_getopt_internal	481
11.85.3.3	exchange	481

11.85.3.4	getopt	481
11.85.3.5	my_index	481
11.85.4	Variable Documentation	481
11.85.4.1	__getopt_initialized	481
11.85.4.2	first_nonopt	481
11.85.4.3	last_nonopt	481
11.85.4.4	nextchar	481
11.85.4.5	optarg	481
11.85.4.6	opterr	481
11.85.4.7	optind	481
11.85.4.8	optopt	481
11.85.4.9	ordering	481
11.85.4.10	posixly_correct	481
11.86	vcc/getopt.h File Reference	481
11.86.1	Define Documentation	483
11.86.1.1	no_argument	483
11.86.1.2	optional_argument	483
11.86.1.3	required_argument	483
11.86.2	Function Documentation	483
11.86.2.1	_getopt_internal	483
11.86.2.2	getopt	483
11.86.2.3	getopt_long	483
11.86.2.4	getopt_long_only	483
11.86.3	Variable Documentation	483
11.86.3.1	optarg	483
11.86.3.2	opterr	483
11.86.3.3	optind	483
11.86.3.4	optopt	483
11.87	vcc/getopt1.c File Reference	483
11.87.1	Define Documentation	484
11.87.1.1	GETOPT_INTERFACE_VERSION	484
11.87.1.2	NULL	484
11.87.2	Function Documentation	484
11.87.2.1	getopt_long	484
11.87.2.2	getopt_long_only	484
11.88	vcc/test-dll/AssemblyInfo.cpp File Reference	484
11.89	vcc/test-dlluse/AssemblyInfo.cpp File Reference	485
11.90	vcc/test-dll/Stdafx.cpp File Reference	485
11.91	vcc/test-dlluse/stdafx.cpp File Reference	486

Chapter 1

SDHLibrary_CPP

The C++ library for controlling the SDH (SCHUNK Dexterous Hand) from a PC

1.1 General project information

Author

Dirk Osswald

Project releases:

- Current release name and changelog: see [PROJECT_RELEASE](#)
- Current release date: see [PROJECT_DATE](#)

1.2 Purpose

This documentation describes the **sdh** library for the C++ language. It allows to control the [SDH](#) (SCHUNK Dexterous Hand) with a user program from a PC.

- For some demonstration programs see: [demonstration programs](#).

1.3 Links

- The SCHUNK homepage: <http://www.schunk.com/>
- Additionally used libraries:
 - The C++ STL (Standard Template Library), especially the `vector<T>` type.
See e.g.:
 - * <http://gcc.gnu.org/onlinedocs/libstdc++/documentation.html>

* [http://msdn.microsoft.com/en-us/library/csc687y\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/csc687y(VS.80).aspx)

- Used tools:

- **gcc** (GNU Compiler Collection), namely g++, the C++ compiler. See <http://gcc.gnu.org/onlinedocs/gcc/>
- **make** (GNU make), see <http://www.gnu.org/software/make/manual/make.html>
- **Microsoft Visual Studio C++** can be used on windows as an alternative to the above
 - * see e.g. [Visual Studio 2008](#)
 - * see e.g. [Visual Studio 2005](#)
- **Doxygen** for generating pdf and html documentation directly from the sources. See <http://www.stack.nl/~dimitri/doxygen/>

1.4 Copyright

Copyright (c) 2007-2011 SCHUNK GmbH & Co. KG

Chapter 2

Bug List

File [demo-GetFingerXYZ.cpp](#) When compiled with MS Visual Studio then using the "-R" command line parameter will make the demonstration program demo-GetFingerXYZ abort.

Class [SDH::cDSA](#) cDSAException: Checksum Error on Windows console While communicating with the tactile sensor controller a "cDSAException: Checksum Error" might be thrown once in a while. This seems to happen only when the program is started from a native windows command console (cmd.exe), but not e.g. when the program is started from a cygwin console. Strange. **Workaround** is to catch the exception and ignore the frame.

=> **Resolved in SDHLibrary-C++ v0.0.2.1**

Problem was an overrun of the windows input buffer, e.g. on heavy processor load.

Member [SDH::cDSA::GetMatrixSensitivity\(int matrix_no\)](#) With DSACON32m firmware R218 and before this did not work, instead the factory default (0.5) was always reported

=> **Resolved in DSACON32m firmware R268**

Member [SDH::cDSA::SetFramerate\(UINT16 framerate, bool do_RLE=true, bool do_data_acquisition=true\)](#) SCHUNK-internal bugzilla ID: Bug 680

With DSACON32m firmware R276 and after and SDHLibrary-C++ v0.0.1.15 and before stopping of the sending did not work.

=> **Resolved in SDHLibrary-C++ v0.0.1.16**

SCHUNK-internal bugzilla ID: Bug 680

With DSACON32m firmware before R276 and SDHLibrary-C++ v0.0.1.16 and before acquiring of single frames did not work

=> **Resolved in SDHLibrary-C++ v0.0.1.17**

SCHUNK-internal bugzilla ID: Bug 703

With DSACON32m firmware R288 and before and SDHLibrary-C++ v0.0.2.1 and before tactile sensor frames could not be read reliably in single frame mode.

=> **Resolved in DSACON32m firmware 2.9.0.0**

=> **Resolved in SDHLibrary-C++ v0.0.2.1 with workaround for older DSACON32m firmwares**

Member `SDH::cRS232::Open(void)` The binary communication introduced in firmware 0.0.2.15 and SDHLibrary-C++ 0.0.2.0 did not work properly on Linux when RS232 was used for communication.

See also:

- [Bug 1011: Bug: Binary communication does not work via RS232 on Linux](#)
 - [SDH_USE_BINARY_COMMUNICATION](#)
- => **Resolved in SDHLibrary 0.0.2.2**

Member `SDH::cSDH::EmergencyStop(void)` For now this will **NOT** work while a GripHand() command is executing, even if that was initiated non-sequentially!

Member `SDH::cSDH::GripHand(eGraspId grip, double close, double velocity, bool sequ=true)`

With SDH firmware > 0.0.2.6 and SDHLibrary < 0.0.1.12 GripHand() does not work ([Bug 575](#))

=> **Resolved in SDHLibrary 0.0.1.12**

With SDH firmware < 0.0.2.6 GripHand() does not work and might yield undefined behaviour there

=> **Resolved in SDH firmware 0.0.2.6**

Currently the performing of a skill or grip with GripHand() can **NOT** be interrupted!!! Even if the command is sent with *sequ=false* it **cannot** be stopped or emergency stopped.

With SDH firmware < 0.0.2.6 GripHand() does not work and might yield undefined behaviour there

=> **Resolved in SDH firmware 0.0.2.6**

Currently the performing of a skill or grip with GripHand() can **NOT** be interrupted!!! Even if the command is sent with *sequ=false* it **cannot** be stopped or emergency stopped.

Member `SDH::cSDH::MoveAxis(std::vector< int >const &axes, bool sequ=true)`

With SDH firmware < 0.0.2.7 calling MoveAxis() while some axes are moving in eCT_POSE controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

Member `SDH::cSDH::MoveFinger(std::vector< int >const &fingers, bool sequ=true)`

With SDH firmware < 0.0.2.7 calling `MoveFinger()` while some axes are moving in `eCT_POSE` controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the `eCT_POSE` controller type with velocity profile `eVP_RAMP`. For the `eCT_POSE` controller type with velocity profile `eVP_SIN_SQUARE` changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

Member `SDH::cSDH::MoveHand(bool sequ=true)` With SDH firmware < 0.0.2.7 calling `MoveHand()` while some axes are moving in `eCT_POSE` controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the `eCT_POSE` controller type with velocity profile `eVP_RAMP`. For the `eCT_POSE` controller type with velocity profile `eVP_SIN_SQUARE` changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

Member `SDH::cSDH::SetAxisValueVector(std::vector< int > const &axes, std::vector< double > const &values, pS`

Setting a single axis velocity only sets the velocities of all other axes to 0! So in order to address only some axes you must provide the desired velocity for all axes that you want to move in every call:

With SDH firmware 0.0.2.16 and `SDHLibrary-CPP` 0.0.2.3 and binary communication (see also [SDH_USE_BINARY_COMMUNICATION](#)) setting of single (more precisely: less-than-all) axis parameters (position, velocity, acceleration) does not work as expected: If a single parameter is to be set for a single axis then that parameter is set to 0 for all other axes.

Workaround: If you want to access several different axes one after another then you have to keep a vector of the parameter for all your used axes in your application. You can then update single values of the vector on demand, but you have to send the complete vector to the `SDHlibrary` functions on every call.

With SDH firmware 0.0.2.16 and `SDHLibrary-CPP` 0.0.2.3 and binary communication (see also [SDH_USE_BINARY_COMMUNICATION](#)) setting of single (more precisely: less-than-all) axis parameters (position, velocity, acceleration) does not work as expected: If a single parameter is to be set for a single axis then that parameter is set to 0 for all other axes.

Workaround: If you want to access several different axes one after another then you have to keep a vector of the parameter for all your used axes in your application. You can then update single values of the vector on demand, but you have to send the complete vector to the `SDHlibrary` functions on every call.

Member `SDH::cSDH::Stop(void)` For now this will **NOT** work while a `GripHand()` command is executing, even if that was initiated non-sequentially!

With SDH firmware < 0.0.2.7 this made the axis jerk in `eCT_POSE` controller type. This is resolved in SDH firmware 0.0.2.7 for the `eCT_POSE` controller type

with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

With SDH firmware < 0.0.2.7 this made the axis jerk in eCT_POSE controller type. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

Member [SDH::cSDH::WaitAxis](#)(std::vector< int > const &axes, double timeout=-1.0)

Due to a bug in SDH firmwares prior to 0.0.2.6 the WaitAxis() command was somewhat unreliable there. When called immediately after a movement command like MoveHand(), then the WaitAxis() command returned immediately without waiting for the end of the movement. With SDH firmwares 0.0.2.6 and newer this is no longer problematic and WaitAxis() works as expected.

=> **Resolved in SDH firmware 0.0.2.6**

With SDH firmware 0.0.2.6 WaitAxis() did not work if one of the new velocity based controllers (eCT_VELOCITY, eCT_VELOCITY_ACCELERATION) was used. With SDH firmwares 0.0.2.7 and newer this now works. Here the WaitAxis() waits until the selected axes come to velocity 0.0

=> **Resolved in SDH firmware 0.0.2.7**

With SDH firmware 0.0.2.6 WaitAxis() did not work if one of the new velocity based controllers (eCT_VELOCITY, eCT_VELOCITY_ACCELERATION) was used. With SDH firmwares 0.0.2.7 and newer this now works. Here the WaitAxis() waits until the selected axes come to velocity 0.0

=> **Resolved in SDH firmware 0.0.2.7**

Member [SDH::cSDHSerial::kv](#)(int axis=All, double *kv=NULL) With [SDH](#) firmware

0.0.2.9 kv() might not respond kv value correctly in case it was changed before. With [SDH](#) firmwares 0.0.2.10 and newer this now works.

=> **Resolved in [SDH](#) firmware 0.0.2.10**

Member [SDH::cSDHSerial::Open](#)(cSerialBase *_com) SCHUNK-internal bugzilla ID: Bug 1517

With [SDH](#) firmware 0.0.3.x the first connection to a newly powered up [SDH](#) can yield an error especially when connecting via TCP.

=> **Resolved in SDHLibrary-C++ 0.0.2.10**

Member [SDH::cSDHSerial::pid](#)(int axis, double *p=NULL, double *i=NULL, double *d=NULL)

With [SDH](#) firmware 0.0.2.9 pid() might not respond pid values correctly in case

these were changed before. With [SDH](#) firmwares 0.0.2.10 and newer this now works.

=> Resolved in [SDH](#) firmware 0.0.2.10

Member [SDH::cSDHSerial::tvav](#)(int axis=All, double *velocity=NULL) With SDH firmware < 0.0.1.0 axis 0 can go no faster than 14 deg/s

=> Resolved in SDH firmware 0.0.1.0

Member [SDH::cSDHSerial::v](#)(int axis=All, double *velocity=NULL) With SDH firmware < 0.0.1.0 axis 0 can go no faster than 14 deg/s

=> Resolved in SDH firmware 0.0.1.0

Member [SDHUSAGE_DEFAULT](#) When compiled with VCC then the macros WITH_ESD_CAN / WITH_PEAK_CAN used above are not available since these are defined in the VCC project settings of the SDHLibrary VCC-Project. Therefore the value of SDHUSAGE_DEFAULT is incorrect and thus the [cSDHOptions](#) will display an incomplete usage string when called with -h/--help.

Workaround: use the online help contained in the doxygen documentation: [Online help of demonstration programs](#)

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Compile time settings	23
Derived compile time settings	24
Demonstration programs	24
Online help of demonstration programs	26

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

SDH	55
-------------------------------	--------------------

Chapter 5

Class Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

SDH::cCANSerial_ESD_Internal	76
SDH::cCANSerial_PEAK_Internal	87
SDH::cCRC	91
SDH::cCRC_DSACON32m	95
SDH::cCRC_SDH	98
SDH::cDBG	101
SDH::cDSA	103
cDSAUpdater	123
SDH::cHexByteString	124
cIsGraspedBase	125
cIsGraspedByArea	128
SDH::cMsg	132
SDH::cSDHBase	224
SDH::cSDH	145
SDH::cSDHSerial	251
SDH::cSDHLibraryException	242
SDH::cDSAException	121
SDH::cSDHErrorCommunication	238
SDH::cSerialBaseException	278
SDH::cCANSerial_ESDException	77
SDH::cCANSerial_PEAKException	88
SDH::cRS232Exception	142
SDH::cRS232Exception	142
SDH::cTCPSerialException	299
SDH::cSDHErrorInvalidParameter	240
SDH::cSimpleVectorException	291
cSDHOptions	245
SDH::cSerialBase	270

SDH::cCANSerial_ESD	69
SDH::cCANSerial_PEAK	80
SDH::cRS232	134
SDH::cRS232	134
SDH::cTCPSerial	293
SDH::cSerialBase::cSetTimeoutTemporarily	280
SDH::cSetValueTemporarily< T >	282
SDH::cSimpleStringList	283
SDH::cSimpleTime	286
SDH::cSimpleVector	288
SDH::cUnitConverter	302
option	306
SDH::cDSA::sContactInfo	307
SDH::cDSA::sControllerInfo	308
SDH::cDSA::sMatrixInfo	309
sRecordedData	311
SDH::cDSA::sResponse	311
SDH::sSDHBinaryRequest	312
SDH::sSDHBinaryResponse	314
SDH::cDSA::sSensitivityInfo	315
SDH::cDSA::sSensorInfo	317
SDH::cDSA::sTactileSensorFrame	317

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

SDH::cCANSerial_ESD (Low-level communication class to access a CAN port from company ESD (http://www.esd.eu/))	69
SDH::cCANSerial_ESD_Internal (Internal hardware specific implementation details of the lowlevel ESD CAN interface)	76
SDH::cCANSerial_ESDException (Derived exception class for low-level CAN ESD related exceptions)	77
SDH::cCANSerial_PEAK (Low-level communication class to access a CAN port from company PEAK (http://www.peak-system.com))	80
SDH::cCANSerial_PEAK_Internal (Internal hardware specific implementation details of the lowlevel PEAK CAN interface)	87
SDH::cCANSerial_PEAKException (Derived exception class for low-level CAN PEAK related exceptions)	88
SDH::cCRC (Cyclic Redundancy Code checker class, used for protecting communication against transmission errors)	91
SDH::cCRC_DSACON32m (A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller)	95
SDH::cCRC_SDH (A derived CRC class that uses a CRC table and initial value suitable for protecting the binary communication with SDH via RS232)	98
SDH::cDBG (A class to print colored debug messages)	101
SDH::cDSA (SDH::cDSA is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH)	103
SDH::cDSAException (Derived exception class for low-level DSA related exceptions)	121
cDSAUpdater (Class to create an updater thread for continuously updating tactile sensor data)	123

SDH::cHexString (Dummy class for (debug) stream output of bytes as list of hex values)	124
cIsGraspedBase (Abstract base class for calculation of IsGrasped condition using tactile sensor information)	125
cIsGraspedByArea (Class for calculation of IsGrasped condition using an expected area of contact measured by the tactile sensors)	128
SDH::cMsg (Class for short, fixed maximum length text messages)	132
SDH::cRS232 (Low-level communication class to access a serial port on Cygwin and Linux)	134
SDH::cRS232Exception (Derived exception class for low-level RS232 related exceptions)	142
SDH::cSDH (SDH::cSDH is the end user interface class to control a SDH (SCHUNK Dexterous Hand))	145
SDH::cSDHBase (The base class to control the SCHUNK Dexterous Hand)	224
SDH::cSDHErrorCommunication (Derived exception class for exceptions related to communication between the SDHLibrary and the SDH)	238
SDH::cSDHErrorInvalidParameter (Derived exception class for exceptions related to invalid parameters)	240
SDH::cSDHLibraryException (Base class for exceptions in the SDHLibrary-CPP)	242
cSDHOptions (Class for command line option parsing holding option parsing results)	245
SDH::cSDHSerial (The class to communicate with a SDH via RS232)	251
SDH::cSerialBase (Low-level communication class to access a serial port)	270
SDH::cSerialBaseException (Derived exception class for low-level serial communication related exceptions)	278
SDH::cSerialBase::cSetTimeoutTemporarily (Helper class to set timeout of <code>_serial_base</code> on construction and reset to previous value on destruction. (RAII-idiom))	280
SDH::cSetValueTemporarily< T > (Helper class to set value on construction and reset to previous value on destruction. (RAII-idiom))	282
SDH::cSimpleStringList (A simple string list. (Fixed maximum number of strings of fixed maximum length))	283
SDH::cSimpleTime (Very simple class to measure elapsed time)	286
SDH::cSimpleVector (A simple vector implementation)	288
SDH::cSimpleVectorException (Derived exception class for low-level simple vector related exceptions)	291
SDH::cTCPSerial (Low-level communication class to access a CAN port)	293
SDH::cTCPSerialException (Derived exception class for low-level CAN ESD related exceptions)	299
SDH::cUnitConverter (Unit conversion class to convert values between physical unit systems)	302
option	306
SDH::cDSA::sContactInfo (Structure to hold info about the contact of one sensor patch)	307
SDH::cDSA::sControllerInfo (A data structure describing the controller info about the remote DSA32m controller)	308
SDH::cDSA::sMatrixInfo (A data structure describing a single sensor matrix connected to the remote DSA32m controller)	309

sRecordedData (Structure to hold current hand state while recording with demo-benchmark)	311
SDH::cDSA::sResponse (Data structure for storing responses from the remote DSACON32m controller)	311
SDH::sSDHBinaryRequest (Data structure with binary data for request from PC to SDH)	312
SDH::sSDHBinaryResponse (Data structure with binary data for response from SDH to PC)	314
SDH::cDSA::sSensitivityInfo (Structure to hold info about the sensitivity settings of one sensor patch)	315
SDH::cDSA::sSensorInfo (A data structure describing the sensor info about the remote DSACON32m controller)	317
SDH::cDSA::sTactileSensorFrame (A data structure describing a full tactile sensor frame read from the remote DSACON32m controller)	317

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Doxyfile (Doxyfile for generating documentation for SDHLibrary cpp using doxygen)	370
Makefile (Makefile for SDH SDHLibrary C project)	371
demo/cancat.cpp (Yet incomplete tool to send and receive data via CAN. See __help__ and online help ("-h" or "--help") for available options)	328
demo/demo-benchmark.cpp (Simple script to benchmark communication speed of the SDH . See __help__ and online help ("-h" or "--help") for available options)	329
demo/demo-contact-grasping.cpp (Simple script to do grasping using tactile sensor info feedback. See __help__ and online help ("-h" or "--help") for available options)	332
demo/demo-dsa-simple.cpp (Simple program to test class cDSA (tactile sensor reading). See __help__ and online help ("-h" or "--help") for available options)	334
demo/demo-dsa.cpp (Simple program to test class cDSA. See __help__ and online help ("-h" or "--help") for available options)	336
demo/demo-GetAxisActualAngle.cpp (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See __help__ and online help ("-h" or "--help") for available options)	338
demo/demo-GetFingerXYZ.cpp (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See __help__ and online help ("-h" or "--help") for available options)	340
demo/demo-griphand.cpp (Very simple demonstration program using the SDHLibrary-CPP: Demonstrate the use of the GripHand command See __help__ and online help ("-h" or "--help") for available options)	342

demo/ demo-mimic.cpp (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See __help__ and online help ("-h" or "--help") for available options)	344
demo/ demo-radians.cpp (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control), commanding in radians. See __help__ and online help ("-h" or "--help") for available options)	346
demo/ demo-simple-withtiming.cpp (Very simple C++ programm to make an attached SDH move)	348
demo/ demo-simple.cpp (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See __help__ and online help ("-h" or "--help") for available options)	350
demo/ demo-simple2.cpp (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Stop. See __help__ and online help ("-h" or "--help") for available options)	352
demo/ demo-simple3.cpp (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and WaitAxis. See __help__ and online help ("-h" or "--help") for available options)	354
demo/ demo-temperature.cpp (Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See __help__ and online help ("-h" or "--help") for available options)	356
demo/ demo-velocity-acceleration.cpp (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger in "velocity with acceleration ramp" control mode. See __help__ and online help ("-h" or "--help") for available options)	358
demo/ dsaboost.cpp (Helper stuff for the "boosted" DSA stuff)	360
demo/ dsaboost.h (Helper stuff for the DSA using boost)	361
demo/ sdhoptions.cpp (Implementation of a class to parse common SDH related command line options)	363
demo/ sdhoptions.h (Implementation of a class to parse common SDH related command line options)	367
sdh/ basisdef.h (This file contains some basic definitions (defines, macros, datatypes))	372
sdh/ canserial-esd.cpp (Implementation of class SDH::cCANSerial_ESD , a class to access an ESD CAN interface on cygwin/linux and Visual Studio)	375
sdh/ canserial-esd.h (Interface of class SDH::cCANSerial_ESD , class to access CAN bus via ESD card on cygwin/linux)	377
sdh/ canserial-peak.cpp (Implementation of class SDH::cCANSerial_PEAK , a class to access a PEAK CAN interface on cygwin/linux and Visual Studio)	379
sdh/ canserial-peak.h (Interface of class SDH::cCANSerial_PEAK , class to access CAN bus via PEAK card on cygwin/linux)	382
sdh/ crc.cpp (Implementation of class SDH::cCRC_DSACON32m (actually only the static members all other is derived))	384

sdh/crc.h (This file contains interface to cCRC, a class to handle CRC calculation)	385
sdh/dbg.h (This file contains interface and implementation of class SDH::cDBG, a class for colorfull debug messages)	387
sdh/dsa.cpp (This file contains definition of SDH::cDSA, a class to communicate with the tactile sensors of the SDH)	389
sdh/dsa.h (This file contains interface to SDH::cDSA, a class to communicate with the tactile sensors of the SDH)	392
sdh/release.h (This file contains nothing but C/C++ defines with the name of the project itself (PROJECT_NAME) and the name of the release (PROJECT_RELEASE) of the whole project)	399
sdh/rs232-cygwin.cpp (Implementation of class SDH::cRS232, a class to access serial RS232 port on cygwin/linux)	414
sdh/rs232-cygwin.h (Interface of class SDH::cRS232, a class to access serial RS232 port on cygwin/linux)	416
sdh/rs232-vcc.cpp (Implementation of class SDH::cRS232, a class to access serial RS232 port with VCC compiler on Windows)	418
sdh/rs232-vcc.h (Implementation of class SDH::cRS232, a class to access serial RS232 port with VCC compiler on Windows)	420
sdh/sdh.cpp (This file contains the interface to class SDH::cSDH, the end user class to access the SDH from a PC)	422
sdh/sdh.h (This file contains the interface to class SDH::cSDH, the end user class to access the SDH from a PC)	424
sdh/sdh_codes.cpp (This file contains function to convert the binary command and return codes of the SDH to strings)	426
sdh/sdh_codes.h (This file contains function to convert the binary command codes of the SDH. To use this from a non gcc compiler you might have to define SDH__attribute__ to nothing and SDH_USE_VCC to 1)	427
sdh/sdh_command_codes.h (This file contains the binary command codes of the SDH. To use this from a non gcc compiler you might have to define SDH__attribute__ to nothing and SDH_USE_VCC to 1)	429
sdh/sdh_return_codes.h (This file contains a typedef for a common Return Codes enum)	434
sdh/sdhbase.cpp (Implementation of class SDH::cSDHBase)	441
sdh/sdhbase.h (Interface of class SDH::cSDHBase)	443
sdh/sdhexception.cpp (Implementation of the exception base class SDH::cSDHLibraryException and SDH::cMsg)	444
sdh/sdhexception.h (Interface of the exception base class SDH::cSDHLibraryException and SDH::cMsg)	446
sdh/sdhlibrary_settings.h (This file contains settings to make the SDHLibrary compile on differen systems:	
• gcc/Cygwin/Windows	
• gcc/Linux	
• VisualC++/Windows	
)	447
sdh/sdhserial.cpp (Interface of class SDH::cSDHSerial)	449
sdh/sdhserial.h (Interface of class SDH::cSDHSerial)	452

sdh/serialbase.cpp (Implementation of class SDH::cSerialBase , a virtual base class to access serial interfaces like RS232 or CAN)	454
sdh/serialbase.h (Interface of class SDH::cSerialBase , a virtual base class to access serial communication channels like RS232 or CAN)	455
sdh/simplestringlist.cpp (Implementation of class SDH::cSimpleStringList)	457
sdh/simplestringlist.h (Interface of class SDH::cSimpleStringList)	459
sdh/simpleteime.h (Interface of auxilliary utility functions for SDHLibrary-CPP)	460
sdh/simplevector.cpp (Implementation of class SDH::cSimpleVector)	462
sdh/simplevector.h (Interface of class SDH::cSimpleVector)	463
sdh/tcpserial.cpp (Implementation of class SDH::cTCPSerial , a class to access a TCP port on cygwin/linux and Visual Studio)	465
sdh/tcpserial.h (Interface of class SDH::cTCPSerial , class to access TCP port cygwin/linux)	466
sdh/unit_converter.cpp (Implementation of class SDH::cUnitConverter)	468
sdh/unit_converter.h (Interface of class SDH::cUnitConverter)	470
sdh/util.cpp (Implementation of auxilliary utility functions for SDHLibrary-CPP)	472
sdh/util.h (Interface of auxilliary utility functions for SDHLibrary-CPP)	474
vcc/getopt.c	478
vcc/getopt.h	481
vcc/getopt1.c	483
vcc/test-dll/AssemblyInfo.cpp	484
vcc/test-dll/Stdafx.cpp	485
vcc/test-dlluse/AssemblyInfo.cpp	485
vcc/test-dlluse/stdafx.cpp	486

Chapter 8

Module Documentation

8.1 Compile time settings

Defines

- `#define SDH_USE_NAMESPACE 1`
Flag, if 1 then all classes are put into a namespace called [SDH](#). If 0 then the classes are left outside any namespace.
- `#define SDH_USE_BINARY_COMMUNICATION 1`
Flag, if 1 then binary communication is used where possible for better performance (requires at least firmware 0.0.2.15)

8.1.1 Detailed Description

Primary settings to be adjusted by the user.

8.1.2 Define Documentation

8.1.2.1 `#define SDH_USE_BINARY_COMMUNICATION 1`

Flag, if 1 then binary communication is used where possible for better performance (requires at least firmware 0.0.2.15)

8.1.2.2 `#define SDH_USE_NAMESPACE 1`

Flag, if 1 then all classes are put into a namespace called [SDH](#). If 0 then the classes are left outside any namespace.

8.2 Derived compile time settings

Defines

- `#define NAMESPACE_SDH_START namespace SDH {`
- `#define NAMESPACE_SDH_END }`
- `#define USING_NAMESPACE_SDH using namespace SDH;`
- `#define NS_SDH SDH::`

8.2.1 Detailed Description

Derived settings that users should not have to adjust. Adjustments should be done in [Compile time settings](#)

8.2.2 Define Documentation

8.2.2.1 `#define NAMESPACE_SDH_END }`

8.2.2.2 `#define NAMESPACE_SDH_START namespace SDH {`

8.2.2.3 `#define NS_SDH SDH::`

8.2.2.4 `#define USING_NAMESPACE_SDH using namespace SDH;`

8.3 Demonstration programs

Files

- file [demo-benchmark.cpp](#)
Simple script to benchmark communication speed of the [SDH](#). See [__help__](#) and online help ("-h" or "--help") for available options.
- file [demo-contact-grasping.cpp](#)
Simple script to do grasping using tactile sensor info feedback. See [__help__](#) and online help ("-h" or "--help") for available options.
- file [demo-dsa-simple.cpp](#)
Simple program to test class `cDSA` (tactile sensor reading). See [__help__](#) and online help ("-h" or "--help") for available options.
- file [demo-dsa.cpp](#)
Simple program to test class `cDSA`. See [__help__](#) and online help ("-h" or "--help") for available options.
- file [demo-GetAxisActualAngle.cpp](#)

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-GetFingerXYZ.cpp](#)

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-griphand.cpp](#)

Very simple demonstration program using the SDHLibrary-CPP: Demonstrate the use of the GripHand command See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-mimic.cpp](#)

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-radians.cpp](#)

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control), commanding in radians. See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-simple.cpp](#)

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-simple2.cpp](#)

Very simple C++ program to make an attached SDH move. With non-sequential call of move and Stop. See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-simple3.cpp](#)

Very simple C++ program to make an attached SDH move. With non-sequential call of move and WaitAxis. See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-temperature.cpp](#)

Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

- file [demo-velocity-acceleration.cpp](#)

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger in "velocity with acceleration ramp" control mode. See [__help__](#) and online help ("-h" or "--help") for available options.

8.3.1 Detailed Description

Some demonstration programs are provided which show the usage of the classes in the SDHLibrary-C++ library:

8.4 Online help of demonstration programs

The provided [demonstration programs](#) have an online help which is shown below:

Online help for program demo-benchmark.exe

```
Simple script to benchmark communication speed of the SDH:
The hand will move to a start position in coordinated position control
mode first. Then periodic movements are performed using the velocity
with acceleration ramp controller while the communication and control
rate is printed.

- Example usage:
  - Make SDH connected via Ethernet move.
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-benchmark --tcp=192.168.1.42:23

  - Make SDH connected to port 2 = COM3 move:
    > demo-benchmark -p 2 --dsaport=3

  - Make SDH connected to USB to RS232 converter 0 move:
    > demo-benchmark --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/ttyUSB1

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The joint controllers and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The SDH controller is attached to the
    default TCP port 23 and the tactile sensors to the default TCP port 13000.
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-benchmark --tcp=192.168.1.42 --dsa_tcp -v

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected to:
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-benchmark --port=2 --dsaport=3 -v
```

```
usage: demo-benchmark [options]
```

General options:

```
-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,
    library (and the attached SDH firmware, if found), then exit.

-V, --version_check
```

Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.

```
-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: CSDH-level messages, 3: CSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from
    SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-c, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program demo-contact-grasping.exe

Simple script to do grasping with tactile sensor info feedback:
The hand will move to a pregrasp pose (open hand). You can then
reach an object to grasp into the hand. The actual grasping

is started as soon as a contact is detected. The finger joints then try to move inwards until a certain force is reached on the corresponding tactile sensors.

- Example usage:
 - Make SDH and tactile sensors connected via Ethernet grasp.
The SDH and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the TCP port 23 and the tactile sensors to TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> demo-contact-grasping --tcp=192.168.1.42:23 --dsa_tcp=:13000
 - Make SDH connected to port 2 = COM3 with tactile sensors connected to port 3 = COM4 grasp:
> demo-contact-grasping -p 2 --dsaport=3
 - Make SDH connected to USB to RS232 converter 0 and with tactile sensors connected to USB to RS232 converter 1 grasp:
> demo-contact-grasping --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/ttyUSB1
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> demo-contact-grasping --tcp=192.168.1.42 --dsa_tcp -v
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected to:
 - port 2 = COM3 (joint controllers) and
 - port 3 = COM4 (tactile sensor controller)
 > demo-contact-grasping --port=2 --dsaport=3 -v

usage: demo-contact-grasping [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program.
Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

```

-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from
    SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-c, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)

DSA options (tactile sensor):
-q PORT, --dsaport=PORT
    use RS232 communication PORT to connect to to tactile sensor controller
    of SDH instead of default 1='COM2'='/dev/ttyS1'.

--dsa_tcp[=[IP_OR_HOSTNAME][:PORT]]
    use TCP for communication with the tactile sensor controller.
    The tactile sensor can be reached via TCP/IP on port PORT at
    IP_OR_HOSTNAME, which can be a numeric IPv4 address or a hostname.
    The default is "192.168.1.42:13000"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.2)

--dsa_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If
    the DEVICE_FORMAT_STRING contains '%d' then the dsa PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

--no_rle

```

Do not use the RunLengthEncoding

Online help for program `demo-dsa-simple.exe`

Online help for program `demo-dsa.exe`

Simple demo to test cDSA class of SDHLibrary-cpp.

Remarks:

- You must specify at least one of these options to see some output:
 - f | --fullframe
 - r | --resulting
 - c | --controllerinfo
 - s | --sensorinfoinfo
 - m | --matrixinfo=N
- Example usage:
 - Read a single full frame from tactile sensors connected to port 3 = COM4:


```
> demo-dsa --dsaport=3 -f
```
 - Read a single full frame from tactile sensors connected via TCP/IP (ethernet)


```
:
```

The tactile sensors have IP-Address 192.168.1.42 and use TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)

```
> demo-dsa --dsa_tcp=192.168.1.42:13000 -f
```
 - Read full frames continuously once per second from tactile sensors connected to port 3 = COM4:


```
> demo-dsa --dsaport=3 -f -r 1
```
 - Read full frames continuously 10 times per second from tactile sensors connected to port 3 = COM4:


```
> demo-dsa --dsaport=3 -f -r 10
```
 - Read full frames continuously as fast as possible (DSA push-mode) from tactile sensors connected to port 3 = COM4:


```
> demo-dsa --dsaport=3 -f -r 30
```
 - Read a single full frame from tactile sensors connected to USB to RS232 converter 0:


```
> demo-dsa --dsa_rs_device=/dev/ttyUSB0 -f
```
 - Read the sensor, controller, matrix 0 infos from tactile sensors connected to port 3 = COM4:


```
> demo-dsa --dsaport=3 -s -c -m 0
```
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)


```
> demo-dsa --tcp=192.168.1.42 --dsa_tcp -v
```

- Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected to
 - port 2 = COM3 (joint controllers) and
 - port 3 = COM4 (tactile sensor controller)

```
> demo-dsa -p 2 --dsaport=3 -v
```
- Set the sensitivity of all tactile sensor matrixes to 0.75 temporarily. The value will be used only temporarily (until reset or power cycle).


```
> demo-dsa --dsaport=3 --sensitivity=0.75
```
- Set the sensitivity of all tactile sensor matrixes to 0.75 persistently. The value will be stored persistently (i.e. will remain after reset or power cycle).


```
> demo-dsa --dsaport=3 --sensitivity=0.75 --persistent
```
- Reset the sensitivity of all tactile sensor matrixes to factory default.


```
> demo-dsa --dsaport=3 --sensitivity=0.75 --reset
```
- Set the sensitivity of tactile sensor matrices 1 and 4 to individual values temporarily. The value will be used only temporarily (until reset or power cycle).
 - Sensor 1 (distal sensor of finger 1) will be set to 0.1
 - Sensor 4 (proximal sensor of finger 3) will be set to 0.4

```
> demo-dsa --dsaport=3 --sensitivity1=0.1 --sensitivity4=0.4
```
- Like for the sensitivity the thresholds can be adjusted via the `--threshold=VALUE` parameter.
- Known bugs:
 - see the bug description for "CDSAException: Checksum Error on Windows console" in the Related Pages->Bug List section of the doxygen documentation

usage: demo-dsa [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program.
Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232

```

        and 1000000 Bit/s (1MBit/s) for CAN

DSA options (tactile sensor):
-q PORT, --dsaport=PORT
    use RS232 communication PORT to connect to tactile sensor controller
    of SDH instead of default l='COM2'='/dev/ttyS1'.

--dsa_tcp=[IP_OR_HOSTNAME][:PORT]
    use TCP for communication with the tactile sensor controller.
    The tactile sensor can be reached via TCP/IP on port PORT at
    IP_OR_HOSTNAME, which can be a numeric IPv4 address or a hostname.
    The default is "192.168.1.42:13000"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.2)

--dsa_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If
    the DEVICE_FORMAT_STRING contains '%d' then the dsa PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

--no_rle
    Do not use the RunLengthEncoding

-r, --framerate=FRAMERATE
    Framerate for acquiring full tactile sensor frames. Default value 0
    means 'acquire a single frame only'. Any value > 0 will make the
    DSACON32m controller in the SDH send data at the highest possible rate
    (ca. 30 FPS (frames per second)).

-f, --fullframe
    Print acquired full frames numerically.

-S, --sensorinfo
    Print sensor info from DSA (texel dimensions, number of texels...).

-C, --controllerinfo
    Print controller info from DSA (version...).

-M, --matrixinfo=MATRIX_INDEX
    Print matrix info for matrix with index MATRIX_INDEX from DSA.
    This includes the current setting for sensitivity and threshold
    of the addressed matrix (if supported by the tactile sensor firmware).
    This option can be used multiple times.
--sensitivity=SENSITIVITY
    Set the sensor sensitivity for all tactile sensor pads to the given
    value. The value SENSITIVITY must be in range [0.0 .. 1.0], where
    0.0 is minimum and 1.0 is maximum sensitivity.
    If --reset is given as well then SENSITIVITY is ignored and
    the sensitivity is reset to the factory default.
    To see the current setting for sensitivity use -M --matrixinfo
    For setting sensitivities individually for a specific sensor X [0..5]
    use --sensitivityX=SENSITIVITY

--sensitivityX=SENSITIVITY
    X is a sensor index in range [0..5]. Set sensor sensitivity for a
    specific sensor X. See also --sensitivity.
    This option can be used multiple times (with different X).

--threshold=THRESHOLD
    Set the sensor threshold for all tactile sensor pads to the given

```

value. The value THRESHOLD must be in range [0 .. 4095], where (0 is minimum, 4095 is maximum threshold).
 If --reset is given as well then THRESHOLD is ignored and the threshold is reset to the factory default.

--thresholdX=THRESHOLD
 Set sensor threshold for a specific sensor X.
 X is a sensor index in range [0..5]. See also option --threshold.
 This option can be used multiple times (with different X).

--reset
 If given, then the values given with --sensitivity(X) and/or --threshold(X) are reset to their factory default.

--persistent
 If given, then all the currently set values for sensitivity and threshold are saved persistently in the configuration memory of the DSACON32m controller in the SDH.
 PLEASE NOTE: the maximum write endurance of the configuration memory is about 100.000 times!

--showdsasettings
 If given, then current settings for sensitivity and threshold will be printed on stdout first.

Online help for program demo-GetAxisActualAngle.exe

Print measured actual axis angles of SDH.
 (C++ demo application using the SDHLibrary-CPP library.)

- Example usage:
 - Print actual angles of an SDH connected via Ethernet.
 The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
 (Requires at least SDH-firmware v0.0.3.1)
 > demo-GetAxisActualAngle --tcp=192.168.1.42:23
 - Print actual angles of an SDH connected to port 2 = COM3 once:
 > demo-GetAxisActualAngle -p 2
 - Print actual angles of an SDH connected to port 2 = COM3 every 500ms:
 > demo-GetAxisActualAngle -p 2 -t 0.5
 - Print actual angles of an SDH connected to USB to RS232 converter 0 once:
 > demo-GetAxisActualAngle --sdh_rs_device=/dev/ttyUSB0
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
 The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
 (Requires at least SDH-firmware v0.0.3.2)
 > demo-GetAxisActualAngle --tcp=192.168.1.42 --dsa_tcp -v
 - Get the version info of an SDH connected to port 2 = COM3
 > demo-GetAxisActualAngle --port=2 -v

usage: demo-GetAxisActualAngle [options]

General options:
 -h, --help

Show this help message and exit.

- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program. Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN
- C, --can, --canesd
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
- n NET, --net=NET
Use ESD CAN NET for CAN communication, default=0.
- canpeak
Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.
- e ID_READ, --id_read=ID_READ
Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).
- w ID_WRITE, --id_write=ID_WRITE
Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).
- tcp[=[IP_OR_HOSTNAME][:PORT]]
Use TCP for communication with the SDH. The SDH can be reached via TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4 address or a hostname. The default is "192.168.1.42:23"
When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME is used for both.
(This feature requires at least SDH firmware 0.0.3.1)

Other options:

- R, --radians
Use radians and radians per second for angles and angular velocities instead of default degrees and degrees per second.
(The demo programs provided might not evaluate this parameter correctly. Thus they might fail if this parameter is used.)
- F, --fahrenheit
Use degrees fahrenheit to report temperatures instead of default degrees celsius.
- t PERIOD, --period=PERIOD
For periodic commands only: Time period of measurements in seconds. The default of '0' means: report once only. If set then the time since start of measurement is printed at the beginning of every line.

Online help for program demo-GetFingerXYZ.exe

Print measured XYZ position of fingertips of SDH.
C++ demo application using the SDHLibrary-CPP library.)

For every finger the actual axis angles and the finger tip coordinates are printed.

- Example usage:
 - Print finger angles and finger tip xyz coordinates of an SDH connected via Ethernet. The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
(Requires at least SDH-firmware v0.0.3.1)
> demo-GetFingerXYZ --tcp=192.168.1.42:23
 - Print finger angles and finger tip xyz coordinates of an SDH connected to port 2 = COM3 once:
> demo-GetFingerXYZ -p 2
 - Print finger angles and finger tip xyz coordinates of an SDH connected to port 2 = COM3 every 500ms:
> demo-GetFingerXYZ -p 2 -t 0.5
 - Print finger angles and finger tip xyz coordinates of an SDH connected to USB to RS232 converter 0 once:
> demo-GetFingerXYZ --sdh_rs_device=/dev/ttyUSB0
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> demo-GetFingerXYZ --tcp=192.168.1.42 --dsa_tcp -v
 - Get the version info of an SDH connected to port 2 = COM3
> demo-GetFingerXYZ --port=2 -v
- Known bugs:
 - Command line parameter "-R" does not work when compiled with MS Visual Studio

usage: demo-GetFingerXYZ [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program.
Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN
- C, --can, --canesd
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
- n NET, --net=NET
Use ESD CAN NET for CAN communication, default=0.
- canpeak
Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.
- e ID_READ, --id_read=ID_READ
Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).
- w ID_WRITE, --id_write=ID_WRITE
Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).
- tcp[=[IP_OR_HOSTNAME][:PORT]]
Use TCP for communication with the SDH. The SDH can be reached via TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4 address or a hostname. The default is "192.168.1.42:23"
When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME

is used for both.
(This feature requires at least SDH firmware 0.0.3.1)

Other options:

- R, --radians
Use radians and radians per second for angles and angular velocities instead of default degrees and degrees per second.
(The demo programs provided might not evaluate this parameter correctly. Thus they might fail if this parameter is used.)
- F, --fahrenheit
Use degrees fahrenheit to report temperatures instead of default degrees celsius.
- t PERIOD, --period=PERIOD
For periodic commands only: Time period of measurements in seconds. The default of '0' means: report once only. If set then the time since start of measurement is printed at the beginning of every line.

Online help for program demo-griphand.exe

Demonstrate the use of the GripHand command.
(C++ demo application using the SDHLibrary-CPP library.)

- Example usage:
 - Make SDH connected via Ethernet move.
The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
(Requires at least SDH-firmware v0.0.3.1)
> demo-griphand --tcp=192.168.1.42:23
 - Make SDH connected to port 2 = COM3 move:
> demo-griphand -p 2
 - Make SDH connected to USB to RS232 converter 0 move:
> demo-griphand --sdh_rs_device=/dev/ttyUSB0
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> demo-griphand --tcp=192.168.1.42 --dsa_tcp -v
 - Get the version info of an SDH connected to port 2 = COM3
> demo-griphand --port=2 -v

usage: demo-griphand [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check

Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.

```
-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT
    Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-C, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program `demo-mimic.exe`

In case you have 2 SDHs you can operate one of them by moving the first hand manually. You must give parameters for 2 hands on the command line. (C++ demo application using the SDHLibrary-CPP library.)

- Example usage:
 - Mimic the manual movements of SDH on port 2 = COM3 with the SDH on port 5 = COM6:


```
> demo-mimic -p 2 -p 5
```
 - Mimic the manual movements of SDH with CAN IDs 0x01 and 0x11 on ESD CAN bus with the SDH with CAN IDs 0x02 and 0x2 on the same ESD CAN bus


```
> demo-mimic --can --id_read 0x1 --id_write 0x11 --can --id_read 0x2 --id_write 0x22
```

usage: demo-mimic [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program. Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN
- c, --can, --canesd
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
- n NET, --net=NET
Use ESD CAN NET for CAN communication, default=0.

```
--canpeak
    Use CAN bus via a PEEK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Other options:

```
-R, --radians
    Use radians and radians per second for angles and angular velocities
    instead of default degrees and degrees per second.
    (The demo programs provided might not evaluate this parameter
    correctly. Thus they might fail if this parameter is used.)

-F, --fahrenheit
    Use degrees fahrenheit to report temperatures instead of default degrees
    celsius.

-t PERIOD, --period=PERIOD
    For periodic commands only: Time period of measurements in seconds. The
    default of '0' means: report once only. If set then the time since start
    of measurement is printed at the beginning of every line.
```

Online help for program demo-radians.exe

Move proximal and distal joints of finger 1 three times by 0.1745 rad (10 degrees).
(C++ demo application using the SDHLibrary-CPP library.)

```
- Example usage:
- Make SDH connected via Ethernet move.
  The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
  (Requires at least SDH-firmware v0.0.3.1)
  > demo-radians --tcp=192.168.1.42:23

- Make SDH connected to port 2 = COM3 move:
  > demo-radians -p 2

- Make SDH connected to USB to RS232 converter 0 move:
  > demo-radians --sdh_rs_device=/dev/ttyUSB0

- Get the version info of both the joint controllers and the tactile
  sensor firmware from an SDH connected via Ethernet.
  The joint controllers and the tactile sensors have a common IP-Address,
  here 192.168.1.42. The SDH controller is attached to the
  default TCP port 23 and the tactile sensors to the default TCP port 13000.
  (Requires at least SDH-firmware v0.0.3.2)
  > demo-radians --tcp=192.168.1.42 --dsa_tcp -v
```

- Get the version info of an SDH connected to port 2 = COM3
> demo-radians --port=2 -v

usage: demo-radians [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program. Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN
- c, --can, --canesd
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
- n NET, --net=NET
Use ESD CAN NET for CAN communication, default=0.
- canpeak
Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.
- e ID_READ, --id_read=ID_READ
Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).
- w ID_WRITE, --id_write=ID_WRITE
Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

```
--tcp=[IP_OR_HOSTNAME][:PORT]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program **demo-ref.exe**

Do reference movement???

usage: demo-ref [options]

General options:

```
-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,
    library (and the attached SDH firmware, if found), then exit.

-V, --version_check
    Check the firmware release of the connected SDH if it is the one
    recommended by this library. A message will be printed accordingly.

-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT
    Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-C, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
```

```
-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program **demo-simple-withtiming.exe**

Move proximal and distal joints of finger 1 three times by 10 degrees and measure time for these actions.
(C++ demo application using the SDHLibrary-CPP library.)

usage: demo-simple-withtiming [options]

General options:

```
-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,
    library (and the attached SDH firmware, if found), then exit.

-V, --version_check
    Check the firmware release of the connected SDH if it is the one
    recommended by this library. A message will be printed accordingly.

-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
```

If the `DEVICE_FORMAT_STRING` contains `'%d'` then the `PORT` must also be provided. If not then the `DEVICE_FORMAT_STRING` is the full device name.

`-T TIMEOUT, --timeout=TIMEOUT` Timeout in seconds when waiting for data from SDH. The default `-1` means: wait forever.

`-b BAUDRATE, --baud=BAUDRATE`
Use `BAUDRATE` in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN

`-C, --can, --canesd`
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

`-n NET, --net=NET`
Use ESD CAN NET for CAN communication, default=0.

`--canpeak`
Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

`-e ID_READ, --id_read=ID_READ`
Use CAN ID `ID_READ` for receiving CAN messages (default: 43=0x2B).

`-w ID_WRITE, --id_write=ID_WRITE`
Use CAN ID `ID_WRITE` for writing CAN messages (default: 42=0x2A).

`--tcp=[IP_OR_HOSTNAME][:PORT]`
Use TCP for communication with the SDH. The SDH can be reached via TCP/IP on port `PORT` at `IP_OR_HOSTNAME`, which can be a numeric IPv4 address or a hostname. The default is "192.168.1.42:23"
When using `--tcp` and `--dsa_tcp` then only the last set `IP_OR_HOSTNAME` is used for both.
(This feature requires at least SDH firmware 0.0.3.1)

Online help for program `demo-simple.exe`

Move proximal and distal joints of finger 1 three times by 10 degrees.
(C++ demo application using the `SDHLibrary-CPP` library.)

- Example usage:

- Make SDH connected via Ethernet move.
The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
(Requires at least SDH-firmware v0.0.3.1)
> `demo-simple --tcp=192.168.1.42:23`
- Make SDH connected to port 2 = COM3 move:
> `demo-simple -p 2`
- Make SDH connected to USB to RS232 converter 0 move:
> `demo-simple --sdh_rs_device=/dev/ttyUSB0`
- Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> `demo-simple --tcp=192.168.1.42 --dsa_tcp -v`
- Get the version info of an SDH connected to port 2 = COM3
> `demo-simple --port=2 -v`

usage: demo-simple [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program. Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN
- c, --can, --canesd
Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.
- n NET, --net=NET
Use ESD CAN NET for CAN communication, default=0.
- canpeak
Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.
- e ID_READ, --id_read=ID_READ
Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).
- w ID_WRITE, --id_write=ID_WRITE
Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).
- tcp[=[IP_OR_HOSTNAME][:PORT]]
Use TCP for communication with the SDH. The SDH can be reached via TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4

address or a hostname. The default is "192.168.1.42:23"
 When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
 is used for both.
 (This feature requires at least SDH firmware 0.0.3.1)

Online help for program demo-simple2.exe

Move proximal and distal joints of finger 1 three times by 10 degrees, stop movement when halfway done.

(C++ demo application using the SDHLibrary-CPP library.)

- Example usage:
 - Make SDH connected via Ethernet move.
 The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
 (Requires at least SDH-firmware v0.0.3.1)
 > demo-simple2 --tcp=192.168.1.42:23
 - Make SDH connected to port 2 = COM3 move:
 > demo-simple2 -p 2
 - Make SDH connected to USB to RS232 converter 0 move:
 > demo-simple2 --sdh_rs_device=/dev/ttyUSB0
 - Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
 The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
 (Requires at least SDH-firmware v0.0.3.2)
 > demo-simple2 --tcp=192.168.1.42 --dsa_tcp -v
 - Get the version info of an SDH connected to port 2 = COM3
 > demo-simple2 --port=2 -v

usage: demo-simple2 [options]

General options:

- h, --help
 Show this help message and exit.
- v, --version
 Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
 Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
 Print debug messages of level LEVEL or lower while executing the program.
 Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
 Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from
    SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-c, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program **demo-simple3.exe**

Move axes 1,2 and 3 to a specific point.
(C++ demo application using the SDHLibrary-CPP library.)

```
- Example usage:
- Make SDH connected via Ethernet move.
  The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
  (Requires at least SDH-firmware v0.0.3.1)
  > demo-simple3 --tcp=192.168.1.42:23

- Make SDH connected to port 2 = COM3 move:
  > demo-simple3 -p 2

- Make SDH connected to USB to RS232 converter 0 move:
  > demo-simple3 --sdh_rs_device=/dev/ttyUSB0

- Get the version info of both the joint controllers and the tactile
  sensor firmware from an SDH connected via Ethernet.
```

The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000. (Requires at least SDH-firmware v0.0.3.2)

```
> demo-simple3 --tcp=192.168.1.42 --dsa_tcp -v
```

```
- Get the version info of an SDH connected to port 2 = COM3
> demo-simple3 --port=2 -v
```

usage: demo-simple3 [options]

General options:

```
-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,
    library (and the attached SDH firmware, if found), then exit.

-V, --version_check
    Check the firmware release of the connected SDH if it is the one
    recommended by this library. A message will be printed accordingly.

-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT
    Timeout in seconds when waiting for data from SDH. The default -1 means:
    wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-C, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.
```

```
-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Online help for program demo-temperature.exe

Print measured temperatures of SDH.
(C++ demo application using the SDHLibrary-CPP library.)

A vector of temperatures is reported. The first 7 temperatures are from sensors close to the corresponding axes motors. The 8th value is the temperature of the FPGA, the controller chip (CPU). The 9th value is the temperature of the PCB (Printed circuit board) in the body of the SDH.

- Example usage:

- Print temperatures of an SDH connected via Ethernet.
The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
(Requires at least SDH-firmware v0.0.3.1)
> demo-GetAxisActualAngle --tcp=192.168.1.42:23
- Print temperatures of an SDH connected to port 2 = COM3 once:
> demo-temperature -p 2
- Print temperatures of an SDH connected to port 2 = COM3 every 500ms:
> demo-temperature -p 2 -t 0.5
- Print temperatures of an SDH connected to USB to RS232 converter 0 move:
> demo-temperature --sdh_rs_device=/dev/ttyUSB0
- Get the version info of both the joint controllers and the tactile sensor firmware from an SDH connected via Ethernet.
The joint controllers and the tactile sensors have a common IP-Address, here 192.168.1.42. The SDH controller is attached to the default TCP port 23 and the tactile sensors to the default TCP port 13000.
(Requires at least SDH-firmware v0.0.3.2)
> demo-GetAxisActualAngle --tcp=192.168.1.42 --dsa_tcp -v
- Get the version info of an SDH connected to port 2 = COM3
> demo-temperature --port=2 -v

usage: demo-temperature [options]

General options:

```
-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,
    library (and the attached SDH firmware, if found), then exit.
```

```
-V, --version_check
    Check the firmware release of the connected SDH if it is the one
    recommended by this library. A message will be printed accordingly.

-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.
```

Communication options:

```
-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT
    Timeout in seconds when waiting for data from SDH. The default -1 means:
    wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-C, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)
```

Other options:

```
-R, --radians
    Use radians and radians per second for angles and angular velocities
    instead of default degrees and degrees per second.
```

(The demo programs provided might not evaluate this parameter correctly. Thus they might fail if this parameter is used.)

- F, --fahrenheit
Use degrees fahrenheit to report temperatures instead of default degrees celsius.
- t PERIOD, --period=PERIOD
For periodic commands only: Time period of measurements in seconds. The default of '0' means: report once only. If set then the time since start of measurement is printed at the beginning of every line.

Online help for program **demo-test.exe**

Tries to connect to SDH, read actual angles and exits.
(C++ demo application using the SDHLibrary-CPP library.)

usage: demo-test [options]

General options:

- h, --help
Show this help message and exit.
- v, --version
Print the version (revision/release names) and dates of application, library (and the attached SDH firmware, if found), then exit.
- V, --version_check
Check the firmware release of the connected SDH if it is the one recommended by this library. A message will be printed accordingly.
- d LEVEL, --debug=LEVEL
Print debug messages of level LEVEL or lower while executing the program.
Level 0 (default): No messages, 1: application-level messages, 2: cSDH-level messages, 3: cSDHSerial-level messages
- l LOGFILE, --debuglog=LOGFILE
Redirect the printed debug messages to LOGFILE instead of default standard error. If LOGFILE starts with '+' then the output will be appended to the file (without the leading '+'), else the file will be overwritten.

Communication options:

- p PORT, --port=PORT, --sdhport=PORT
Use RS232 communication PORT to connect to the SDH instead of the default 0='COM1'='/dev/ttyS0'.
- sdh_rs_device=DEVICE_FORMAT_STRING
Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d". If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be provided. If not then the DEVICE_FORMAT_STRING is the full device name.
- T TIMEOUT, --timeout=TIMEOUT
Timeout in seconds when waiting for data from SDH. The default -1 means: wait forever.
- b BAUDRATE, --baud=BAUDRATE
Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232 and 1000000 Bit/s (1MBit/s) for CAN

```

-c, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp=[IP_OR_HOSTNAME][:PORT]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)

```

Online help for program **demo-velocity-acceleration.exe**

Make the SDH move one finger in "velocity with acceleration ramp" control mode.
(C++ demo application using the SDHLibrary-CPP library.)

```

- Make SDH connected via Ethernet move.
  The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
  (Requires at least SDH-firmware v0.0.3.1)
  > demo-velocity-acceleration --tcp=192.168.1.42:23

- Example usage:
  - Make SDH connected to port 2 = COM3 move:
    > demo-velocity-acceleration -p 2

  - Make SDH connected to USB to RS232 converter 0 move:
    > demo-velocity-acceleration --sdh_rs_device=/dev/ttyUSB0

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The joint controllers and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The SDH controller is attached to the
    default TCP port 23 and the tactile sensors to the default TCP port 13000.
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-velocity-acceleration --tcp=192.168.1.42 --dsa_tcp -v

  - Get the version info of an SDH connected to port 2 = COM3
    > demo-velocity-acceleration --port=2 -v

```

usage: demo-velocity-acceleration [options]

General options:

```

-h, --help
    Show this help message and exit.

-v, --version
    Print the version (revision/release names) and dates of application,

```



```

    library (and the attached SDH firmware, if found), then exit.

-V, --version_check
    Check the firmware release of the connected SDH if it is the one
    recommended by this library. A message will be printed accordingly.

-d LEVEL, --debug=LEVEL
    Print debug messages of level LEVEL or lower while executing the program.
    Level 0 (default): No messages, 1: application-level messages,
    2: cSDH-level messages, 3: cSDHSerial-level messages

-l LOGFILE, --debuglog=LOGFILE
    Redirect the printed debug messages to LOGFILE instead of default
    standard error. If LOGFILE starts with '+' then the output will be
    appended to the file (without the leading '+'), else the file will be
    overwritten.

Communication options:

-p PORT, --port=PORT, --sdhport=PORT
    Use RS232 communication PORT to connect to the SDH instead of the default
    0='COM1'='/dev/ttyS0'.

--sdh_rs_device=DEVICE_FORMAT_STRING
    Use DEVICE_FORMAT_STRING instead of the default "/dev/ttyS%d". Useful
    e.g. to use USB to RS232 converters available via "/dev/ttyUSB%d".
    If the DEVICE_FORMAT_STRING contains '%d' then the PORT must also be
    provided. If not then the DEVICE_FORMAT_STRING is the full device name.

-T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from
    SDH. The default -1 means: wait forever.

-b BAUDRATE, --baud=BAUDRATE
    Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232
    and 1000000 Bit/s (1MBit/s) for CAN

-c, --can, --canesd
    Use CAN bus via an ESD adapter to connect to the SDH instead of RS232.

-n NET, --net=NET
    Use ESD CAN NET for CAN communication, default=0.

--canpeak
    Use CAN bus via a PEAK adapter to connect to the SDH instead of RS232.

-e ID_READ, --id_read=ID_READ
    Use CAN ID ID_READ for receiving CAN messages (default: 43=0x2B).

-w ID_WRITE, --id_write=ID_WRITE
    Use CAN ID ID_WRITE for writing CAN messages (default: 42=0x2A).

--tcp[=[IP_OR_HOSTNAME][:PORT]]
    Use TCP for communication with the SDH. The SDH can be reached via
    TCP/IP on port PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
    address or a hostname. The default is "192.168.1.42:23"
    When using --tcp and --dsa_tcp then only the last set IP_OR_HOSTNAME
    is used for both.
    (This feature requires at least SDH firmware 0.0.3.1)

```


Chapter 9

Namespace Documentation

9.1 SDH Namespace Reference

Classes

- class [cCANSerial_ESD_Internal](#)
internal hardware specific implementation details of the lowlevel ESD CAN interface
- class [cCANSerial_ESDException](#)
Derived exception class for low-level CAN ESD related exceptions.
- class [cCANSerial_ESD](#)
Low-level communication class to access a CAN port from company ESD (<http://www.esd.eu/>)
- class [cCANSerial_PEAK_Internal](#)
internal hardware specific implementation details of the lowlevel PEAK CAN interface
- class [cCANSerial_PEAKException](#)
Derived exception class for low-level CAN PEAK related exceptions.
- class [cCANSerial_PEAK](#)
Low-level communication class to access a CAN port from company PEAK (<http://www.peak-system.com>)
- class [cCRC](#)
Cyclic Redundancy Code checker class, used for protecting communication against transmission errors.
- class [cCRC_DSACON32m](#)
A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller.

- class [cCRC_SDH](#)
A derived CRC class that uses a CRC table and initial value suitable for protecting the binary communication with [SDH](#) via RS232.
- class [cDBG](#)
A class to print colored debug messages.
- class [cHexByteString](#)
dummy class for (debug) stream output of bytes as list of hex values
- class [cDSAException](#)
Derived exception class for low-level DSA related exceptions.
- class [cDSA](#)
[SDH::cDSA](#) is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH.
- class [cRS232Exception](#)
Derived exception class for low-level RS232 related exceptions.
- class [cRS232](#)
Low-level communication class to access a serial port on Cygwin and Linux.
- class [cSDH](#)
[SDH::cSDH](#) is the end user interface class to control a SDH (SCHUNK Dexterous Hand).
- class [cSDHErrorInvalidParameter](#)
Derived exception class for exceptions related to invalid parameters.
- class [cSDHBase](#)
The base class to control the SCHUNK Dexterous Hand.
- class [cMsg](#)
Class for short, fixed maximum length text messages.
- class [cSDHLibraryException](#)
Base class for exceptions in the SDHLibrary-CPP.
- class [cSDHErrorCommunication](#)
Derived exception class for exceptions related to communication between the SDHLibrary and the SDH.
- struct [sSDHBinaryRequest](#)
data structure with binary data for request from PC to [SDH](#)
- struct [sSDHBinaryResponse](#)

data structure with binary data for response from [SDH](#) to PC

- class [cSDHSerial](#)
The class to communicate with a SDH via RS232.
- class [cSerialBaseException](#)
Derived exception class for low-level serial communication related exceptions.
- class [cSerialBase](#)
Low-level communication class to access a serial port.
- class [cSimpleStringList](#)
A simple string list. (Fixed maximum number of strings of fixed maximum length)
- class [cSimpleTime](#)
Very simple class to measure elapsed time.
- class [cSimpleVectorException](#)
Derived exception class for low-level simple vector related exceptions.
- class [cSimpleVector](#)
A simple vector implementation.
- class [cTCPSerialException](#)
Derived exception class for low-level CAN ESD related exceptions.
- class [cTCPSerial](#)
Low-level communication class to access a CAN port.
- class [cUnitConverter](#)
Unit conversion class to convert values between physical unit systems.
- class [cSetValueTemporarily](#)
*helper class to set value on construction and reset to previous value on destruction.
(RAII-idiom)*

Typedefs

- typedef int8_t [Int8](#)
signed integer, size 1 Byte (8 Bit)
- typedef uint8_t [UInt8](#)
unsigned integer, size 1 Byte (8 Bit)
- typedef int16_t [Int16](#)

signed integer, size 2 Byte (16 Bit)

- typedef uint16_t [UInt16](#)

unsigned integer, size 2 Byte (16 Bit)

- typedef int32_t [Int32](#)

signed integer, size 4 Byte (32 Bit)

- typedef uint32_t [UInt32](#)

unsigned integer, size 4 Byte (32 Bit)

- typedef void * [PCAN_HANDLE](#)

Linux libpcan uses HANDLE where Windows Pcan_usb.h uses no handle at all:

- typedef [UInt16](#) [tCRCValue](#)

the data type used to calculate and exchange CRC values with DSACON32m (16 bit integer)

- typedef [cSimpleVector](#)([cSDHSerial::*](#) [pSetFunction](#))(int, double *)

Type of a pointer to a "set-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).

- typedef [cSimpleVector](#)([cSDHSerial::*](#) [pGetFunction](#))(int, double *)

Type of a pointer to a "get-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).

- typedef void * [tDeviceHandle](#)

generic device handle for CAN devices

- typedef time_t [tTimevalSec](#)

- typedef suseconds_t [tTimevalUsec](#)

- typedef double([cUnitConverter::*](#) [pDoubleUnitConverterFunction](#))(double) const

Type of a pointer to a function like 'double [SDH::cUnitConverter::ToExternal](#)(double) const' or 'double [SDH::cUnitConverter::ToInternal](#)(double) const'.

Enumerations

- enum { [eNUMBER_OF_ELEMENTS](#) = [cSimpleVector::eNUMBER_OF_ELEMENTS](#) }

Functions

- [VCC_EXPORT](#) std::ostream & [operator<<](#) (std::ostream &stream, [cHexByteString](#) const &s)

output the bytes in s to stream as a list of space separated hex bytes (without 0x prefix)

- `std::ostream & operator<< (std::ostream &stream, cDSA::sResponse const &response)`
- `VCC_EXPORT std::ostream & operator<< (std::ostream &stream, cDSA::sControllerInfo const &controller_info)`
- `VCC_EXPORT std::ostream & operator<< (std::ostream &stream, cDSA::sSensorInfo const &sensor_info)`
- `VCC_EXPORT std::ostream & operator<< (std::ostream &stream, cDSA::sMatrixInfo const &matrix_info)`
- `VCC_EXPORT std::ostream & operator<< (std::ostream &stream, cDSA const &dsa)`
- `char const * SDHCommandCodeToString (eCommandCode cc)`
- `char const * SDHReturnCodeToString (eReturnCode rc)`
- `std::ostream & operator<< (std::ostream &stream, cMsg const &msg)`
- `std::ostream & operator<< (std::ostream &stream, cSDHLibraryException const &e)`
- `struct SDH::sSDHBinaryRequest __attribute__((packed))`
- `std::ostream & operator<< (std::ostream &stream, sSDHBinaryRequest const &request)`

helper functions to insert a human readable form of the request into stream

- `std::ostream & operator<< (std::ostream &stream, sSDHBinaryResponse const &response)`

helper functions to insert a human readable form of the restore into stream

- `std::ostream & operator<< (std::ostream &stream, cSimpleStringList const &ssl)`

Output of cSimpleStringList objects in 'normal' output streams.

- `std::vector< int > NumerifyRelease (char const *rev)`

Auxiliary functions

- `bool InIndex (int v, int max)`
- `bool InRange (double v, double min, double max)`
- `bool InRange (int n, double const *v, double const *min, double const *max)`
- `double ToRange (double v, double min, double max)`
- `void ToRange (int n, double *v, double const *min, double const *max)`
- `void ToRange (std::vector< double > &v, std::vector< double > const &min, std::vector< double > const &max)`
- `void ToRange (cSimpleVector &v, std::vector< double > const &min, std::vector< double > const &max)`
- `double Approx (double a, double b, double eps)`
- `bool Approx (int n, double *a, double *b, double *eps)`
- `double DegToRad (double d)`
- `double RadToDeg (double r)`
- `void SleepSec (double t)`
- `int CompareReleases (char const *rev1, char const *rev2)`

compare release strings

- `template<typename Function , typename Tp >`
`void apply (Function f, Tp &sequence)`
- `template<typename Function , typename InputIterator >`
`Function apply (Function f, InputIterator first, InputIterator last)`
- `template<typename Function , typename Tp >`
`Tp map (Function f, Tp sequence)`
- `template<typename T >`
`std::ostream & operator<< (std::ostream &stream, std::vector< T > const &v)`

Variables

- `SDH::cRS232Exception SDH__attribute__`
- `std::ostream * g_sdh_debug_log = &std::cerr`
- `cUnitConverter const uc_identity ("any","any","?", 1.0, 0.0, 4)`
Identity converter (internal = external)

9.1.1 Detailed Description

A namespace for all classes and functions in the SDHLibrary.

The use of the namespace can be disabled at compile time of the library by setting [SDH_USE_NAMESPACE](#) to 0.

9.1.2 Typedef Documentation

9.1.2.1 `typedef int16_t SDH::Int16`

signed integer, size 2 Byte (16 Bit)

9.1.2.2 `typedef int32_t SDH::Int32`

signed integer, size 4 Byte (32 Bit)

9.1.2.3 `typedef int8_t SDH::Int8`

signed integer, size 1 Byte (8 Bit)

9.1.2.4 `typedef void* SDH::PCAN_HANDLE`

Linux libpcan uses HANDLE where Windows Pcan_usb.h uses no handle at all:

9.1.2.5 `typedef double(cUnitConverter::* SDH::pDoubleUnitConverterFunction)(double) const`

Type of a pointer to a function like 'double [SDH::cUnitConverter::ToExternal\(double \) const](#)' or 'double [SDH::cUnitConverter::ToInternal\(double \) const](#)'.

9.1.2.6 `typedef cSimpleVector(cSDHSerial::* SDH::pGetFunction)(int, double *)`

Type of a pointer to a "get-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).

9.1.2.7 `typedef cSimpleVector(cSDHSerial::* SDH::pSetFunction)(int, double *)`

Type of a pointer to a "set-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).

9.1.2.8 `typedef UInt16 SDH::tCRCValue`

the data type used to calculate and exchange CRC values with DSACON32m (16 bit integer)

9.1.2.9 `typedef void* SDH::tDeviceHandle`

generic device handle for CAN devices

9.1.2.10 `typedef time_t SDH::tTimevalSec`

9.1.2.11 `typedef suseconds_t SDH::tTimevalUSec`

9.1.2.12 `typedef uint16_t SDH::UInt16`

unsigned integer, size 2 Byte (16 Bit)

9.1.2.13 `typedef uint32_t SDH::UInt32`

unsigned integer, size 4 Byte (32 Bit)

9.1.2.14 `typedef uint8_t SDH::UInt8`

unsigned integer, size 1 Byte (8 Bit)

9.1.3 Enumeration Type Documentation

9.1.3.1 anonymous enum

Enumerator:

eNUMBER_OF_ELEMENTS

9.1.4 Function Documentation

9.1.4.1 struct SDH::sSDHBinaryResponse SDH::_attribute_ ((packed))

9.1.4.2 template<typename Function , typename Tp > void SDH::apply (Function *f*, Tp & *sequence*)

Apply a function to every element of a sequence, the elements of the sequence are modified by *f*

Parameters

<i>f</i>	A unary function object.
<i>sequence</i>	The iterable sequence to modify.

Applies the function object \mathfrak{f} to each element of the *sequence*.

9.1.4.3 template<typename Function , typename InputIterator > Function SDH::apply (Function *f*, InputIterator *first*, InputIterator *last*)

Apply a function to every element of a sequence.

Parameters

<i>first</i>	An input iterator.
<i>last</i>	An input iterator.
<i>f</i>	A unary function object.

Returns

\mathfrak{f} .

Applies the function object \mathfrak{f} to each element in the range [first,last). \mathfrak{f} must not modify the order of the sequence. If \mathfrak{f} has a return value it is ignored.

9.1.4.4 VCC_EXPORT double SDH::Approx (double *a*, double *b*, double *eps*)

Return True if *a* is approximately the same as *b*. I.E. $|a-b| < \text{eps}$

9.1.4.5 VCC_EXPORT bool SDH::Approx (int *n*, double * *a*, double * *b*, double * *eps*)

Return True if list/tuple/array $a=(a_1,a_2,\dots)$ is approximately the same as $b=(b_1,b_2,\dots)$.
I.E. $|a_i-b_i| < \text{eps}[i]$

9.1.4.6 VCC_EXPORT int SDH::CompareReleases (char const * *rev1*, char const * *rev2*)

compare release strings

compare release strings *rev1* and *rev2*.

Returns

-1,0, or 1 if *rev1* is older, equal or newer than *rev2*

Parameters

<i>rev1</i>	- a release string like "0.0.1.5" or "0.0.1.11-a"
<i>rev2</i>	- another release string

Example:

- CompareReleases("0.0.1.5", "0.0.1.5") ==> 0
- CompareReleases("0.0.1.5", "0.0.1.4") ==> 1
- CompareReleases("0.0.1.5", "0.0.2.1") ==> -1
- CompareReleases("0.0.1.5", "0.0.1.5-a") ==> -1

9.1.4.7 VCC_EXPORT double SDH::DegToRad (double *d*)

Return *d* in deg converted to rad

9.1.4.8 VCC_EXPORT bool SDH::InIndex (int *v*, int *max*)

Return True if *v* is in range [0 .. *max*]

9.1.4.9 VCC_EXPORT bool SDH::InRange (int *n*, double const * *v*, double const * *min*, double const * *max*)

Return True if in list/tuple/array $v=(v_1,v_2,\dots)$ each v_i is in range $[\text{min}_i..\text{max}_i]$ with
 $\text{min} = (\text{min}_1, \text{min}_2,\dots)$ $\text{max} = (\text{max}_1, \text{max}_2, ..)$

9.1.4.10 VCC_EXPORT bool SDH::InRange (double *v*, double *min*, double *max*)

Return True if *v* is in range [min .. *max*]

9.1.4.11 `template<typename Function , typename Tp > Tp SDH::map (Function f, Tp sequence)`

map a function to every element of a sequence, returning a copy with the mapped elements

Parameters

<i>f</i>	- A unary function object.
<i>sequence</i>	- An iterable object.

Returns

copy of *sequence* with the mapped elements

9.1.4.12 `std::vector<int> SDH::NumerifyRelease (char const * rev)`

return a vector of integer numbers for a release string *rev*

Parameters

<i>rev</i>	release string like "0.0.1.11-a"
------------	----------------------------------

Returns

a vector of integer numbers like [0,0,1,11,1]

9.1.4.13 `std::ostream & SDH::operator<< (std::ostream & stream, sSDHBinaryRequest const & request)`

helper functions to insert a human readable form of the *request* into *stream*

9.1.4.14 `VCC_EXPORT std::ostream& SDH::operator<< (std::ostream & stream, cHexString const & s) [inline]`

output the bytes in *s* to *stream* as a list of space separated hex bytes (without 0x prefix)

9.1.4.15 `template<typename T > std::ostream& SDH::operator<< (std::ostream & stream, std::vector< T > const & v)`

Overloaded insertion operator for vectors: a comma and space separated list of the vector elements of *v* is inserted into *stream*

Parameters

<i>stream</i>	- the output stream to insert into
<i>v</i>	- the vector of objects to insert into <i>stream</i>

Returns

the stream with the inserted objects

Attention

If you use the [SDH](#) namespace then you should be aware that using this overloaded insertion operator can get tricky:

- If you use a `using namespace SDH` directive then things are easy and intuitive:

```
#include <sdh/util.h>
using namespace SDH;
std::vector<int> v;
std::cout << "this is a std::vector: " << v << "\n";
```

- But without the `using namespace SDH` accessing the operator is tricky, you have to use the 'functional' access `operator<<(s, v)` in order to be able to apply the scope resolution operator `::` correctly:

```
#include <sdh/util.h>
std::vector<int> v;
SDH::operator<<( std::cout << "this is a std::vector: ", v ) << "\n";
```

- The more intuitive approaches do not work:

```
std::cout << faa ; // obviously fails with "no match for >>op
erator<<<< in >>std::cout << faa<<", as expected
std::cout SDH::<< faa ; // is a syntax error
std::cout SDH::operator<< faa ; // is a syntax error
std::cout operator SDH::<< faa ; // is a syntax error
```

9.1.4.16 `std::ostream & SDH::operator<< (std::ostream & stream, cDSA::sSensorInfo const & sensor_info)`

9.1.4.17 `VCC_EXPORT std::ostream & SDH::operator<< (std::ostream & stream, cMsg const & msg)`

9.1.4.18 `VCC_EXPORT std::ostream & SDH::operator<< (std::ostream & stream, cSDHLibraryException const & e)`

9.1.4.19 `VCC_EXPORT std::ostream & SDH::operator<< (std::ostream & stream, cSimpleStringList const & ssl)`

Output of [cSimpleStringList](#) objects in 'normal' output streams.

9.1.4.20 `std::ostream & SDH::operator<< (std::ostream & stream, cDSA::sMatrixInfo const & matrix_info)`

9.1.4.21 `std::ostream & SDH::operator<< (std::ostream & stream, sSDHBinaryResponse const & response)`

helper functions to insert a human readable form of the *restore* into *stream*

9.1.4.22 `VCC_EXPORT std::ostream & SDH::operator<< (std::ostream & stream,
cDSA::sResponse const & response)`

9.1.4.23 `std::ostream & SDH::operator<< (std::ostream & stream, cDSA const & dsa)`

9.1.4.24 `std::ostream & SDH::operator<< (std::ostream & stream, cDSA::sControllerInfo
const & controller_info)`

9.1.4.25 `VCC_EXPORT double SDH::RadToDeg (double r)`

Return *r* in rad converted to deg

9.1.4.26 `char const * SDH::SDHCommandCodeToString (eCommandCode cc)`

9.1.4.27 `char const * SDH::SDHReturnCodeToString (eReturnCode rc)`

9.1.4.28 `VCC_EXPORT void SDH::SleepSec (double t)`

Sleep for *t* seconds. (*t* is a double!)

9.1.4.29 `VCC_EXPORT void SDH::ToRange (int n, double * v, double const * min, double
const * max)`

Limit each *v_i* in *v* to range [*min_i*..*max_i*] with *min* = (*min*1, *min*2,...) *max* = (*max*1, *max*2, ..) This modifies **v*!

9.1.4.30 `VCC_EXPORT double SDH::ToRange (double v, double min, double max)`

Return *v* limited to range [*min* .. *max*]. I.e. if *v* is < *min* then *min* is returned, or if *v* > *max* then *max* is returned, else *v* is returned

9.1.4.31 `VCC_EXPORT void SDH::ToRange (std::vector< double > & v, std::vector< double
> const & min, std::vector< double > const & max)`

Limit each *v_i* in *v* to range [*min_i*..*max_i*] with *min* = (*min*1, *min*2,...) *max* = (*max*1, *max*2, ..) This modifies *v*!

9.1.4.32 `VCC_EXPORT void SDH::ToRange (cSimpleVector & v, std::vector< double > const &
min, std::vector< double > const & max)`

Limit each *v_i* in *v* to range [*min_i*..*max_i*] with *min* = (*min*1, *min*2,...) *max* = (*max*1, *max*2, ..) This modifies *v*!

9.1.5 Variable Documentation

9.1.5.1 VCC_EXPORT std::ostream * SDH::g_sdh_debug_log = &std::cerr

9.1.5.2 SDH::cSDHErrorInvalidParameter SDH::SDH__attribute__

9.1.5.3 cUnitConverter const SDH::uc_identity

Identity converter (internal = external)

Chapter 10

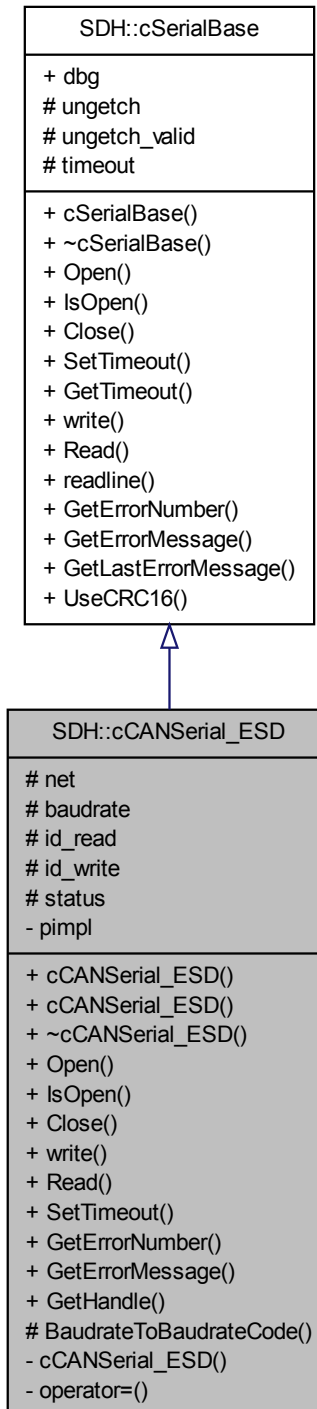
Class Documentation

10.1 SDH::cCANSerial_ESD Class Reference

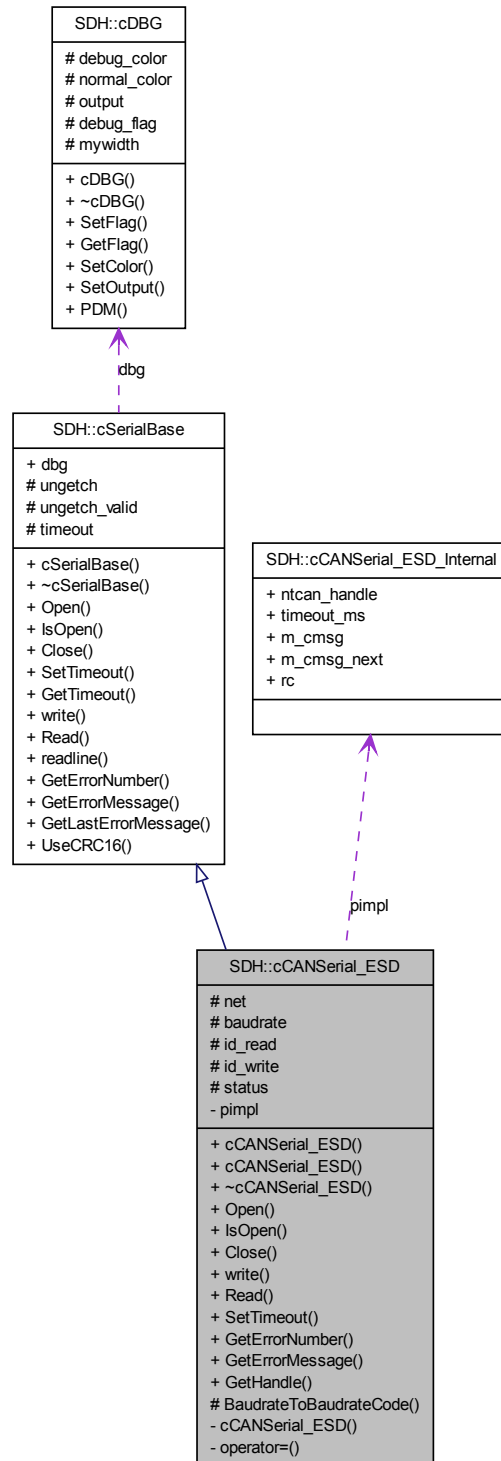
Low-level communication class to access a CAN port from company ESD (<http://www.esd.eu/>)

```
#include <canserial-esd.h>
```

Inheritance diagram for SDH::cCANSerial_ESD:



Collaboration diagram for SDH::cCANSerial_ESD:



Public Member Functions

- [cCANSerial_ESD](#) (int _net, unsigned long _baudrate, double _timeout, int _id_read, int _id_write) throw (cCANSerial_ESDException*)
- [cCANSerial_ESD](#) (tDeviceHandle _ntcan_handle, double _timeout, int _id_read, int _id_write) throw (cCANSerial_ESDException*)
- [~cCANSerial_ESD](#) ()
destructor: clean up
- void [Open](#) (void) throw (cCANSerial_ESDException*)
- bool [IsOpen](#) (void) throw ()
Return true if interface to CAN ESD is open.
- void [Close](#) (void) throw (cCANSerial_ESDException*)
Close the previously opened CAN ESD interface port.
- int [write](#) (char const *ptr, int len=0) throw (cCANSerial_ESDException*)
Write data to a previously opened port.
- ssize_t [Read](#) (void *data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cCANSerial_ESDException*)
- void [SetTimeout](#) (double _timeout) throw (cSerialBaseException*)
set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- virtual tErrorCode [GetErrorNumber](#) ()
- virtual char const * [GetErrorMessage](#) (tErrorCode dw)
- tDeviceHandle [GetHandle](#) ()
return the internally used NTCAN_HANDLE cast to a tDeviceHandle

Protected Member Functions

- unsigned int [BaudrateToBaudrateCode](#) (unsigned long baudrate) throw (cCANSerial_ESDException*)
Translate a baudrate given as unsigned long into a baudrate code for struct termios.

Protected Attributes

- int [net](#)
the ESD CAN net to use
- unsigned long [baudrate](#)
the baudrate to use in bit/s

- int [id_read](#)
the CAN ID used for reading
- int [id_write](#)
the CAN ID used for writing
- int [status](#)

10.1.1 Detailed Description

Low-level communication class to access a CAN port from company ESD (<http://www.esd.eu/>) Since SDHLibrary-C++ release 0.0.2.0 implementation specific parts of the access to ESD CAN devices have been removed from the header file here in order to get rid of dependencies from the weird ntc.h. Specifically the ntc_handle of type NTCAN_HANDLE member was removed. You can still provide an existing handle for reuse on construction, but you must cast your NTCAN_HANDLE to a tDeviceHandle. You can get the internally used NTCAN_HANDLE cast to a tDeviceHandle with [GetHandle\(\)](#).

10.1.2 Constructor & Destructor Documentation

10.1.2.1 cCANSerial_ESD::cCANSerial_ESD (int *_net*, unsigned long *_baudrate*, double *_timeout*, int *_id_read*, int *_id_write*) throw (cCANSerial_ESDException*)

Constructor: constructs an object to communicate with an SDH via CAN bus using an ESD CAN card.

Parameters

<i>_net</i>	- the ESD CAN net to use
<i>_baudrate</i>	- the baudrate in bit/s. Only some bitrates are valid: (1000000,800000,500000,250000,125000,100000,50000,20000,10000)
<i>_timeout</i>	- the timeout in seconds (0 for no timeout = wait for ever)
<i>_id_read</i>	- the CAN ID to use for reading (The SDH sends data on this ID)
<i>_id_write</i>	- the CAN ID to use for writing (The SDH receives data on this ID)

10.1.2.2 cCANSerial_ESD::cCANSerial_ESD (tDeviceHandle *_ntcan_handle*, double *_timeout*, int *_id_read*, int *_id_write*) throw (cCANSerial_ESDException*)

Constructor: constructs an object to communicate with an SDH via CAN bus using an ESD CAN card by reusing an already existing handle.

Parameters

<i>_ntcan_handle</i>	- the ESD CAN handle to reuse (please cast your NTCAN_HANDLE to tDeviceHandle! It is save to do that!)
<i>_timeout</i>	- the timeout in seconds (0 for no timeout = wait for ever)
<i>_id_read</i>	- the CAN ID to use for reading (The SDH sends data on this ID)
<i>_id_write</i>	- the CAN ID to use for writing (The SDH receives data on this ID)

10.1.2.3 cCANSerial_ESD::~~cCANSerial_ESD ()

destructor: clean up

10.1.3 Member Function Documentation**10.1.3.1 unsigned int cCANSerial_ESD::BaudrateToBaudrateCode (unsigned long *baudrate*)
throw (cCANSerial_ESDException*) [protected]**

Translate a baudrate given as unsigned long into a baudrate code for struct termios.

**10.1.3.2 void cCANSerial_ESD::Close (void) throw (cCANSerial_ESDException*)
[virtual]**

Close the previously opened CAN ESD interface port.

Implements [SDH::cSerialBase](#).

**10.1.3.3 char const * cCANSerial_ESD::GetErrorMessage (tErrorCode *dw*)
[virtual]**

Overloaded helper function that returns an ESD error message for ESD error code dw.

Remarks

The string returned will be overwritten by the next call to the function

Reimplemented from [SDH::cSerialBase](#).

10.1.3.4 cSerialBase::tErrorCode cCANSerial_ESD::GetErrorNumber () [virtual]

Overloaded helper function that returns the last ESD error number.

Reimplemented from [SDH::cSerialBase](#).

10.1.3.5 tDeviceHandle cCANSerial_ESD::GetHandle ()

return the internally used NTCAN_HANDLE cast to a tDeviceHandle

10.1.3.6 bool cCANSerial_ESD::IsOpen (void) throw () [virtual]

Return true if interface to CAN ESD is open.

Implements [SDH::cSerialBase](#).

10.1.3.7 `void cCANSerial_ESD::Open (void) throw (cCANSerial_ESDException*)`
[virtual]

Open the device as configured by the parameters given to the constructor

Implements [SDH::cSerialBase](#).

10.1.3.8 `ssize_t cCANSerial_ESD::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cCANSerial_ESDException*)` [virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implements [SDH::cSerialBase](#).

10.1.3.9 `void cCANSerial_ESD::SetTimeout (double _timeout) throw (cSerialBaseException*)` [virtual]

set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

Reimplemented from [SDH::cSerialBase](#).

10.1.3.10 `int cCANSerial_ESD::write (char const * ptr, int len = 0) throw (cCANSerial_ESDException*)` [virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the CAN device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

Implements [SDH::cSerialBase](#).

10.1.4 Member Data Documentation

10.1.4.1 `unsigned long SDH::cCANSerial_ESD::baudrate` [protected]

the baudrate to use in bit/s

10.1.4.2 int SDH::cCANSerial_ESD::id_read [protected]

the CAN ID used for reading

10.1.4.3 int SDH::cCANSerial_ESD::id_write [protected]

the CAN ID used for writing

10.1.4.4 int SDH::cCANSerial_ESD::net [protected]

the ESD CAN net to use

10.1.4.5 int SDH::cCANSerial_ESD::status [protected]

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

- [sdh/canserial-esd.h](#)
- [sdh/canserial-esd.cpp](#)

10.2 SDH::cCANSerial_ESD Internal Class Reference

internal hardware specific implementation details of the lowlevel ESD CAN interface

Public Attributes

- NTCAN_HANDLE [ntcan_handle](#)
the internal handle to the driver
- int32_t [timeout_ms](#)
- CMSG [m_cmsg](#)
- int [m_cmsg_next](#)
index of next received data byte to return to user in m_cmsg
- NTCAN_RESULT [rc](#)

10.2.1 Detailed Description

internal hardware specific implementation details of the lowlevel ESD CAN interface
private data of [cCANSerial_ESD](#).

Required to keep inclusion of [ntcan.h](#) out of public interface of [cCANSerial_ESD](#).

10.2.2 Member Data Documentation

10.2.2.1 MSG SDH::cCANSerial_ESD_Internal::m_msg

received messages might be split over several CAN messages it might therefore happen that more data is received than can be returned to the user. To not loose that data it is kept here to be returned in a later call

10.2.2.2 int SDH::cCANSerial_ESD_Internal::m_msg_next

index of next received data byte to return to user in m_msg

10.2.2.3 NTCAN_HANDLE SDH::cCANSerial_ESD_Internal::ntcan_handle

the internal handle to the driver

10.2.2.4 NTCAN_RESULT SDH::cCANSerial_ESD_Internal::rc

10.2.2.5 int32_t SDH::cCANSerial_ESD_Internal::timeout_ms

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

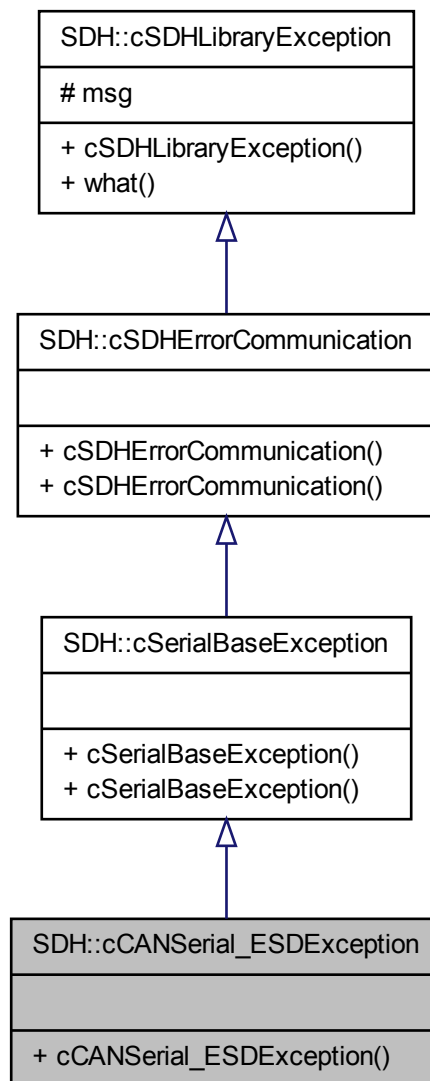
- [sdh/canserial-esd.cpp](#)

10.3 SDH::cCANSerial_ESDException Class Reference

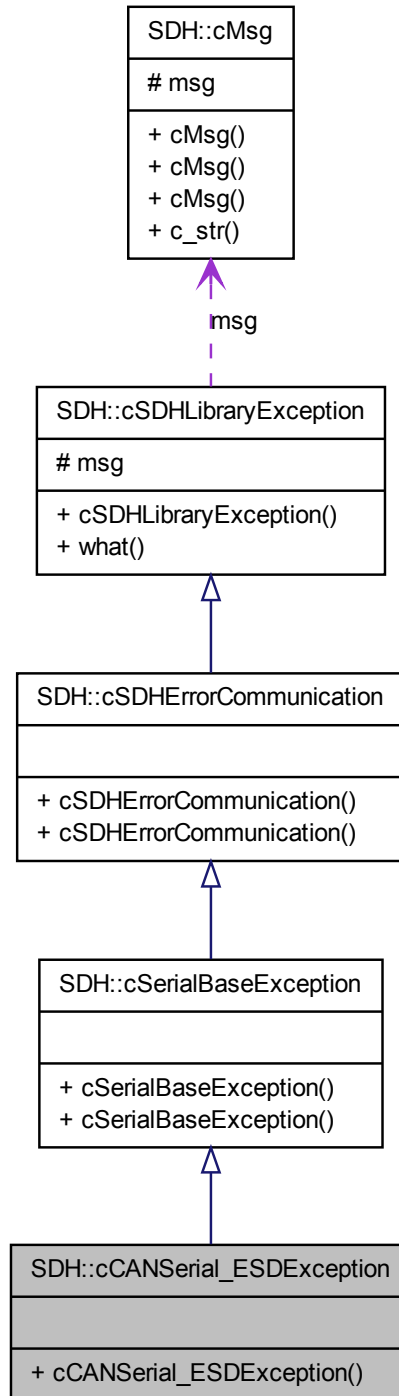
Derived exception class for low-level CAN ESD related exceptions.

```
#include <canserial-esd.h>
```

Inheritance diagram for SDH::cCANSerial_ESDException:



Collaboration diagram for SDH::cCANSerial_ESDException:



Public Member Functions

- [cCANSerial_ESDException](#) (cMsg const &_msg)

10.3.1 Detailed Description

Derived exception class for low-level CAN ESD related exceptions.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 SDH::cCANSerial_ESDException::cCANSerial_ESDException (cMsg const & _msg)
[inline]

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

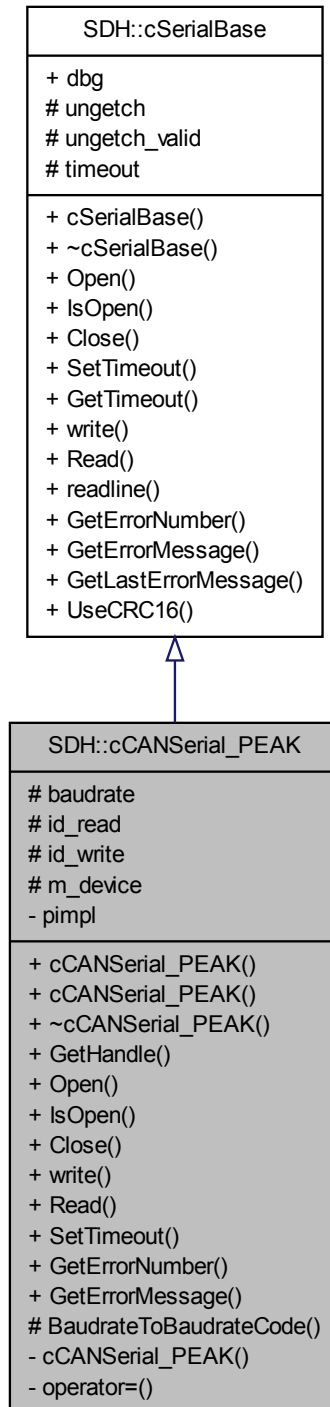
- [sdh/canserial-esd.h](#)

10.4 SDH::cCANSerial_PEAK Class Reference

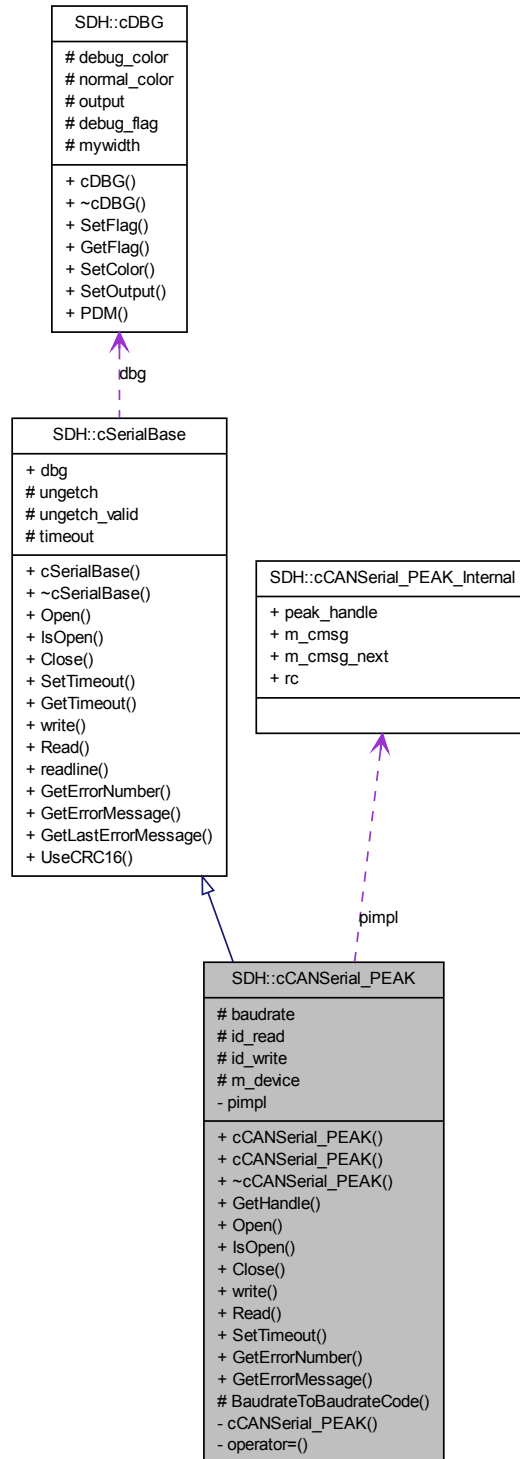
Low-level communication class to access a CAN port from company PEAK (<http://www.peak-system.com>)

```
#include <canserial-peak.h>
```

Inheritance diagram for SDH::cCANSerial_PEAK:



Collaboration diagram for SDH::cCANSerial_PEAK:



Public Member Functions

- [cCANSerial_PEAK](#) (unsigned long _baudrate, double _timeout, int _id_read, int _id_write, const char *device="/dev/pcanusb0") throw (cCANSerial_PEAKEException*)
- [cCANSerial_PEAK](#) (tDeviceHandle _peak_handle, double _timeout, int _id_read, int _id_write) throw (cCANSerial_PEAKEException*)
- [~cCANSerial_PEAK](#) ()
destructor: clean up
- [tDeviceHandle GetHandle](#) ()
- void [Open](#) (void) throw (cCANSerial_PEAKEException*)
- bool [IsOpen](#) (void) throw ()
Return true if interface to CAN PEAK is open.
- void [Close](#) (void) throw (cCANSerial_PEAKEException*)
Close the previously opened CAN PEAK interface port.
- int [write](#) (char const *ptr, int len=0) throw (cCANSerial_PEAKEException*)
Write data to a previously opened port.
- ssize_t [Read](#) (void *data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cCANSerial_PEAKEException*)
- void [SetTimeout](#) (double _timeout) throw (cSerialBaseException*)
set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- virtual [tErrorCode GetErrorNumber](#) ()
- virtual char const * [GetErrorMessage](#) (tErrorCode dw)

Protected Member Functions

- int [BaudrateToBaudrateCode](#) (unsigned long baudrate) throw (cCANSerial_PEAKEException*)
Translate a baudrate given as unsigned long into a baudrate code for struct termios.

Protected Attributes

- unsigned long [baudrate](#)
the baudrate to use in bit/s
- int [id_read](#)
the CAN ID used for reading
- int [id_write](#)
the CAN ID used for writing
- char [m_device](#) [64]

10.4.1 Detailed Description

Low-level communication class to access a CAN port from company PEAK (<http://www.peak-system.com>). Since SDHLibrary-C++ release 0.0.2.0 implementation specific parts of the access to PEAK CAN devices have been removed from the header file here in order to get rid of dependencies from the Pcan_usb.h. Specifically the `peak_handle` of type `PEAK_HANDLE` member was removed. You can still provide an existing handle for reuse on construction, but you must cast your `HANDLE` to a `tDeviceHandle`. You can get the internally used `PEAK_HANDLE` cast to a `tDeviceHandle` with [GetHandle\(\)](#).

10.4.2 Constructor & Destructor Documentation

10.4.2.1 `cCANSerial_Peak::cCANSerial_Peak (unsigned long _baudrate, double _timeout, int _id_read, int _id_write, const char * device = "/dev/pcanusb0") throw (cCANSerial_PeakException*)`

Constructor: constructs an object to communicate with an [SDH](#) via CAN bus using a PEAK CAN card.

Parameters

<i>_baudrate</i>	- the baudrate in bit/s. Only some bitrates are valid: (1000000,800000,500000,250000,125000,100000,50000,20000,10000)
<i>_timeout</i>	- the timeout in seconds (0 for no timeout = wait for ever)
<i>_id_read</i>	- the CAN ID to use for reading (The SDH sends data on this ID)
<i>_id_write</i>	- the CAN ID to use for writing (The SDH receives data on this ID)
<i>device</i>	- the name of the char device to communicate with the PEAD driver (Needed on Linux only!)

10.4.2.2 `cCANSerial_Peak::cCANSerial_Peak (tDeviceHandle _peak_handle, double _timeout, int _id_read, int _id_write) throw (cCANSerial_PeakException*)`

Constructor: constructs an object to communicate with an [SDH](#) via CAN bus using a PEAK CAN card by reusing an already existing handle (will work in Linux only).

Parameters

<i>_peak_handle</i>	- the PEAK CAN handle to reuse (Works on Linux only!)
<i>_timeout</i>	- the timeout in seconds (0 for no timeout = wait for ever)
<i>_id_read</i>	- the CAN ID to use for reading (The SDH sends data on this ID)
<i>_id_write</i>	- the CAN ID to use for writing (The SDH receives data on this ID)

10.4.2.3 `cCANSerial_Peak::~cCANSerial_Peak ()`

destructor: clean up

10.4.3 Member Function Documentation

10.4.3.1 `int cCANSerial_PEAK::BaudrateToBaudrateCode (unsigned long baudrate) throw (cCANSerial_PEAKException*)` [protected]

Translate a baudrate given as unsigned long into a baudrate code for struct termios.

10.4.3.2 `void cCANSerial_PEAK::Close (void) throw (cCANSerial_PEAKException*)`
[virtual]

Close the previously opened CAN PEAK interface port.

Implements [SDH::cSerialBase](#).

10.4.3.3 `char const * cCANSerial_PEAK::GetErrorMessage (tErrorCode dw)`
[virtual]

Overloaded helper function that returns a PEAK error message for error code dw.

Remarks

The string returned will be overwritten by the next call to the function

Reimplemented from [SDH::cSerialBase](#).

10.4.3.4 `cSerialBase::tErrorCode cCANSerial_PEAK::GetErrorNumber ()`
[virtual]

Overloaded helper function that returns the last Peak error number.

Reimplemented from [SDH::cSerialBase](#).

10.4.3.5 `tDeviceHandle cCANSerial_PEAK::GetHandle ()`

Return the value of the HANDLE to the actual CAN device. Works on Linux only!
Only returns a valid handle after a call to [Open\(\)](#)!

Remarks

The returned handle can be used to open a connection to a second [SDH](#) on the same CAN bus

Returns

the handle to the actual CAN device

10.4.3.6 `bool cCANSerial_PEAK::IsOpen (void) throw ()` [virtual]

Return true if interface to CAN PEAK is open.

Implements [SDH::cSerialBase](#).

10.4.3.7 `void cCANSerial_PEAK::Open (void) throw (cCANSerial_PEAKException*)` [virtual]

Open the device as configured by the parameters given to the constructor

Implements [SDH::cSerialBase](#).

10.4.3.8 `ssize_t cCANSerial_PEAK::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cCANSerial_PEAKException*)` [virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implements [SDH::cSerialBase](#).

10.4.3.9 `void cCANSerial_PEAK::SetTimeout (double timeout) throw (cSerialBaseException*)` [virtual]

set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

Reimplemented from [SDH::cSerialBase](#).

10.4.3.10 `int cCANSerial_PEAK::write (char const * ptr, int len = 0) throw (cCANSerial_PEAKException*)` [virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the CAN device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

Implements [SDH::cSerialBase](#).

10.4.4 Member Data Documentation

10.4.4.1 `unsigned long SDH::cCANSerial_PEAK::baudrate` [protected]

the baudrate to use in bit/s

10.4.4.2 `int SDH::cCANSerial_PEAK::id_read` [protected]

the CAN ID used for reading

10.4.4.3 `int SDH::cCANSerial_PEAK::id_write` [protected]

the CAN ID used for writing

10.4.4.4 `char SDH::cCANSerial_PEAK::m_device[64]` [protected]

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

- [sdh/canserial-peak.h](#)
- [sdh/canserial-peak.cpp](#)

10.5 SDH::cCANSerial_PEAK_Internal Class Reference

internal hardware specific implementation details of the lowlevel PEAK CAN interface

Public Attributes

- [PCAN_HANDLE peak_handle](#)
the internal handle to the driver
- [TPCANMsg m_cmsg](#)
- `int m_cmsg_next`
index of next received data byte to return to user in m_cmsg
- `DWORD rc`
last return code of calls to Peak functions

10.5.1 Detailed Description

internal hardware specific implementation details of the lowlevel PEAK CAN interface
private data of [cCANSerial_PEAK](#).

Required to keep inclusion of [Pcan_usb.h](#) out of public interface of [cCANSerial_PEAK](#).

10.5.2 Member Data Documentation

10.5.2.1 TPCANMsg SDH::cCANSerial_PEAK_Internal::m_cmsg

received messages might be split over several CAN messages it might therefore happen that more data is received than can be returned to the user. To not loose that data it is kept here to be returned in a later call

10.5.2.2 int SDH::cCANSerial_PEAK_Internal::m_cmsg_next

index of next received data byte to return to user in m_cmsg

10.5.2.3 PCAN_HANDLE SDH::cCANSerial_PEAK_Internal::peak_handle

the internal handle to the driver

10.5.2.4 DWORD SDH::cCANSerial_PEAK_Internal::rc

last return code of calls to Peak functions

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

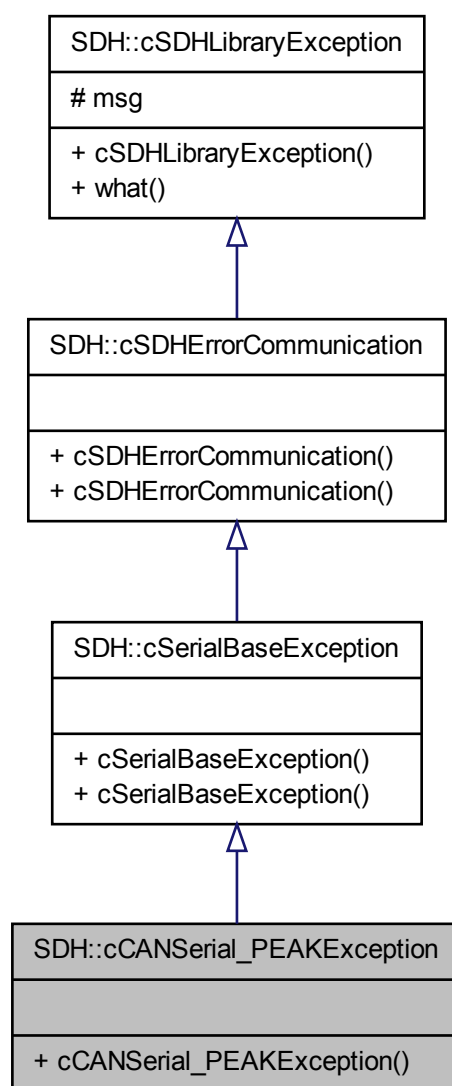
- [sdh/canserial-peak.cpp](#)

10.6 SDH::cCANSerial_PEAKException Class Reference

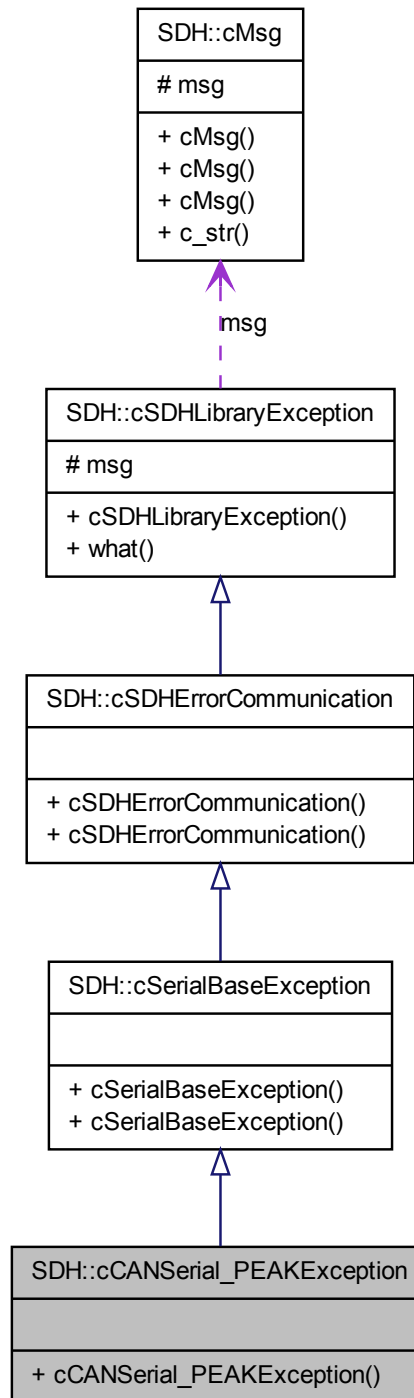
Derived exception class for low-level CAN PEAK related exceptions.

```
#include <canserial-peak.h>
```

Inheritance diagram for SDH::cCANSerial_PEAKEException:



Collaboration diagram for SDH::cCANSerial_PEAKEException:



Public Member Functions

- [cCANSerial_PEAKException](#) ([cMsg](#) const &_msg)

10.6.1 Detailed Description

Derived exception class for low-level CAN PEAK related exceptions.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 SDH::cCANSerial_PEAKException::cCANSerial_PEAKException ([cMsg](#) const & *msg*
) `[inline]`

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

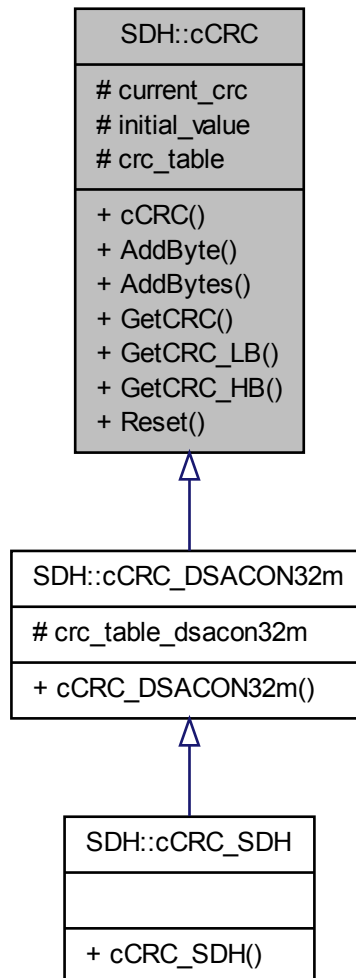
- [sdh/canserial-peak.h](#)

10.7 SDH::cCRC Class Reference

Cyclic Redundancy Code checker class, used for protecting communication against transmission errors.

```
#include <crc.h>
```

Inheritance diagram for SDH::cCRC:



Public Member Functions

- `cCRC` (`tCRCValue` const *`_crc_table`, `tCRCValue` `_initial_value`)
constructor: create a new `cCRC` object and initialize the current value of the CRC checksum. `crc_table` is the CRC table to use.
- `tCRCValue` `AddByte` (unsigned char byte)

insert byte into CRC calculation and return the new current CRC checksum

- [tCRCValue AddBytes](#) (unsigned char *bytes, int nb_bytes)
insert nb_bytes from bytes into CRC calculation and return the new current CRC checksum
- [tCRCValue GetCRC](#) ()
return the current CRC value
- [UInt8 GetCRC_LB](#) ()
return the low byte of the current CRC value
- [UInt8 GetCRC_HB](#) ()
return the high byte of the current CRC value
- [tCRCValue Reset](#) ()
reset the current CRC value to its initial value and return it;

Protected Attributes

- [tCRCValue current_crc](#)
current value of the CRC checksum
- [tCRCValue initial_value](#)
initial value of the CRC checksum
- [tCRCValue const * crc_table](#)
table with precalculated CRC values

10.7.1 Detailed Description

Cyclic Redundancy Code checker class, used for protecting communication against transmission errors. Generic class to calculate a CRC using a given, precalculated table.

Use derived classes like [cCRC_DSACON32m](#) with a specifically set CRC table.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 [SDH::cCRC::cCRC](#) ([tCRCValue const * _crc_table](#), [tCRCValue _initial_value](#))
[inline]

constructor: create a new [cCRC](#) object and initialize the current value of the CRC checksum. *crc_table* is the CRC table to use.

10.7.3 Member Function Documentation

10.7.3.1 `tCRCValue SDH::cCRC::AddByte (unsigned char byte)` `[inline]`

insert byte into CRC calculation and return the new current CRC checksum

10.7.3.2 `tCRCValue SDH::cCRC::AddBytes (unsigned char * bytes, int nb.bytes)`
`[inline]`

insert *nb_bytes* from *bytes* into CRC calculation and return the new current CRC checksum

10.7.3.3 `tCRCValue SDH::cCRC::GetCRC ()` `[inline]`

return the current CRC value

10.7.3.4 `UInt8 SDH::cCRC::GetCRC_HB ()` `[inline]`

return the high byte of the current CRC value

10.7.3.5 `UInt8 SDH::cCRC::GetCRC_LB ()` `[inline]`

return the low byte of the current CRC value

10.7.3.6 `tCRCValue SDH::cCRC::Reset ()` `[inline]`

reset the current CRC value to its initial value and return it;

10.7.4 Member Data Documentation

10.7.4.1 `tCRCValue const* SDH::cCRC::crc_table` `[protected]`

table with precalculated CRC values

10.7.4.2 `tCRCValue SDH::cCRC::current_crc` `[protected]`

current value of the CRC checksum

10.7.4.3 `tCRCValue SDH::cCRC::initial_value` `[protected]`

initial value of the CRC checksum

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

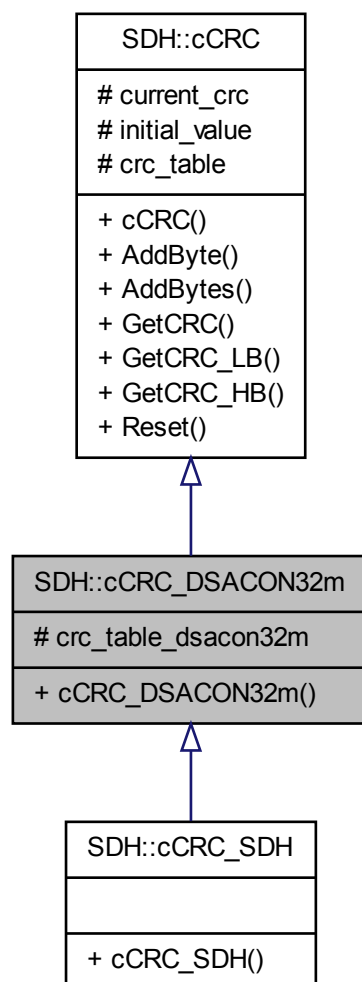
- [sdh/crc.h](#)

10.8 SDH::cCRC_DSACON32m Class Reference

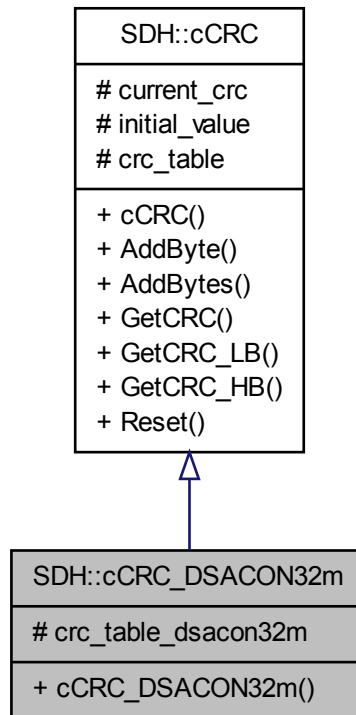
A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller.

```
#include <crc.h>
```

Inheritance diagram for SDH::cCRC_DSACON32m:



Collaboration diagram for SDH::cCRC_DSACON32m:



Public Member Functions

- [cCRC_DSACON32m](#) (void)

constructor to create a [cCRC](#) object suitable for checksumming the communication with a DSACON32m tactile sensor controller

Static Protected Attributes

- static [tCRCValue](#) const [crc_table_dsacon32m](#) [256]

the CRC table used by the DSACON32m controller

10.8.1 Detailed Description

A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 SDH::cCRC_DSACON32m::cCRC_DSACON32m(void) [inline]

constructor to create a [cCRC](#) object suitable for checksumming the communication with a DSACON32m tactile sensor controller

10.8.3 Member Data Documentation

10.8.3.1 tCRCValue const cCRC_DSACON32m::crc_table_dsacon32m [static, protected]

Initial value:

```
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcdbc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xaba1,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbfb1, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
}
```

the CRC table used by the DSACON32m controller

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

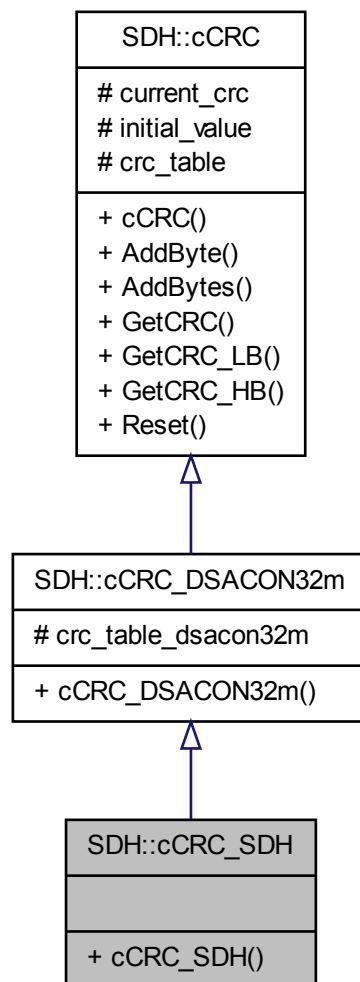
- [sdh/crc.h](#)
- [sdh/crc.cpp](#)

10.9 SDH::cCRC_SDH Class Reference

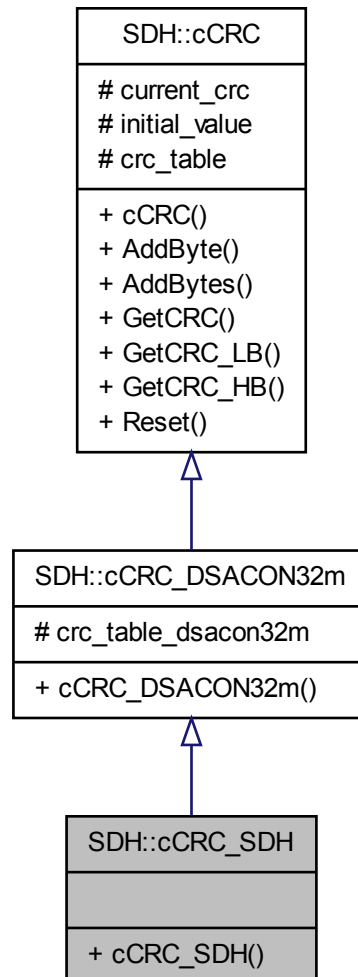
A derived CRC class that uses a CRC table and initial value suitable for protecting the binary communication with [SDH](#) via RS232.

```
#include <crc.h>
```

Inheritance diagram for SDH::cCRC_SDH:



Collaboration diagram for SDH::cCRC_SDH:



Public Member Functions

- [`cCRC_SDH`](#) (void)

constructor to create a [`cCRC`](#) object suitable for checksumming the binary communication with [`SDH`](#)

10.9.1 Detailed Description

A derived CRC class that uses a CRC table and initial value suitable for protecting the binary communication with [SDH](#) via RS232. (for now we use the same CRC as for DSACON32m, but we use this separate class to simplify future changes).

10.9.2 Constructor & Destructor Documentation

10.9.2.1 SDH::cCRC_SDH::cCRC_SDH (void) [inline]

constructor to create a [cCRC](#) object suitable for checksumming the binary communication with [SDH](#)

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

- [sdh/crc.h](#)

10.10 SDH::cDBG Class Reference

A class to print colored debug messages.

```
#include <dbg.h>
```

Public Member Functions

- [cDBG](#) (bool flag=false, char const *color="red", std::ostream *fd=&std::cerr)
- [~cDBG](#) ()
- void [SetFlag](#) (bool flag)
- bool [GetFlag](#) (void) const
- void [SetColor](#) (char const *color)
- void [SetOutput](#) (std::ostream *fd)
- void [PDM](#) (char const *fmt,...) [SDH__attribute__](#)((format(printf

Protected Attributes

- char const * [debug_color](#)
- char const * [normal_color](#)
- std::ostream * [output](#)
- bool [debug_flag](#)
- std::streamsize [mywidth](#)

10.10.1 Detailed Description

A class to print colored debug messages.

- The printing can be switched on or off so the debug code can remain in the code. (default is off)
- The messages can be colorized (default is red).
- The output can be redirected. (default is sys.stderr)
- Debug messages can be printed in a functional way or in C++ stream like way

If the environment variable "SDH_NO_COLOR" is defined then the messages are printed without coloring (usefull for logging or if your terminal does not support colors).

Example:

```
#include "sdh/dbg.h"
d = cDBG( true );
g = cDBG( true, "green" );

d.PDM( "This message is printed in default color red" );
g << "and this one in a nice green ";

g << "of course you can debug print any objects that have a string representat
ion: " << 08 << 15 << true;

g << "Messages can be turned of and on, e.g. selected by command line options"
;
g.SetFlag(false);
g << "This messages is not printed";
```

10.10.2 Constructor & Destructor Documentation

10.10.2.1 `SDH::cDBG::cDBG (bool flag = false, char const * color = "red", std::ostream * fd = &std::cerr) [inline]`

constructor: construct a `cDBG` object

Parameters

<i>flag</i>	- the initial state of the flag, if true then messages sent to the object are printed. Default is false. Can be changed with SetFlag()
<i>color</i>	- the name of the color to use, default is "red". Can be changed with SetColor()
<i>fd</i>	- the ostream to use for output, default is stderr. Can be changed with SetOutput()

10.10.2.2 `SDH::cDBG::~cDBG () [inline]`

10.10.3 Member Function Documentation

10.10.3.1 `bool SDH::cDBG::GetFlag (void) const [inline]`

Get debug_flag of this `cDBG` object.

10.10.3.2 void SDH::cDBG::PDM (char const * *fmt*, ...)

Print debug messages of printf like *fmt*, ... in the color set with `SetColor`, but only if `debug_flag` is true.

10.10.3.3 void SDH::cDBG::SetColor (char const * *color*) [inline]

Set `debug_color` of this `cDBG` object to *color*. *color* is a string like "red", see `util.py` for valid names.

Attention

The string is **NOT** copied, just a pointer is stored

10.10.3.4 void SDH::cDBG::SetFlag (bool *flag*) [inline]

Set `debug_flag` of this `cDBG` object to *flag*. After setting the flag to true debug messages are printed, else not.

10.10.3.5 void SDH::cDBG::SetOutput (std::ostream * *fd*) [inline]

Set output of this `cDBG` object to *fd*, which must be a file like object like `sys.stderr`

10.10.4 Member Data Documentation

10.10.4.1 char const* SDH::cDBG::debug_color [protected]

10.10.4.2 bool SDH::cDBG::debug_flag [protected]

10.10.4.3 std::streamsize SDH::cDBG::mywidth [protected]

10.10.4.4 char const* SDH::cDBG::normal_color [protected]

10.10.4.5 std::ostream* SDH::cDBG::output [protected]

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

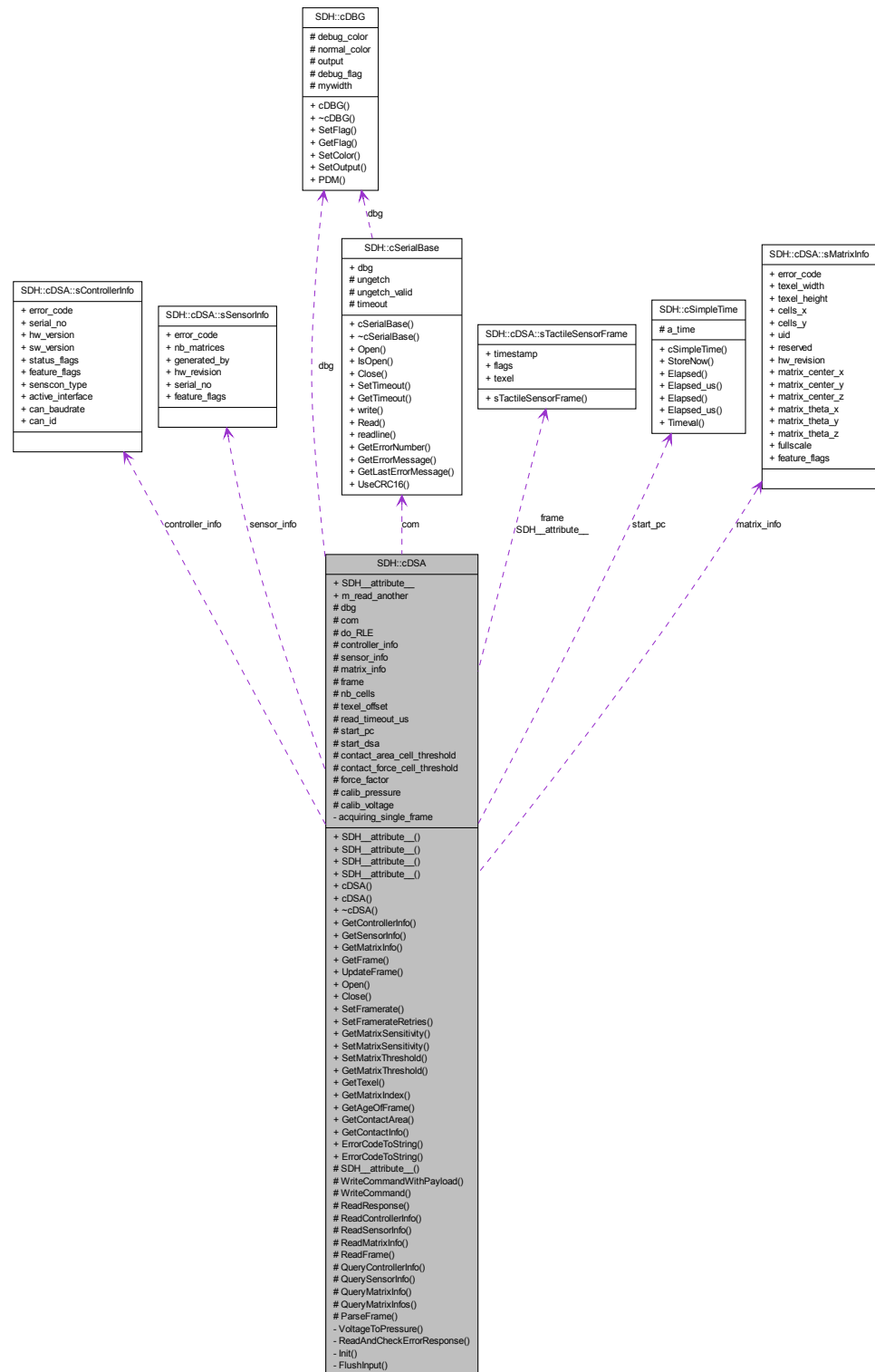
- [sdh/dbg.h](#)

10.11 SDH::cDSA Class Reference

`SDH::cDSA` is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH.

```
#include <dsa.h>
```

Collaboration diagram for SDH::cDSA:



Classes

- struct [sContactInfo](#)
Structure to hold info about the contact of one sensor patch.
- struct [sControllerInfo](#)
A data structure describing the controller info about the remote DSACON32m controller.
- struct [sMatrixInfo](#)
A data structure describing a single sensor matrix connected to the remote DSACON32m controller.
- struct [sResponse](#)
data structure for storing responses from the remote DSACON32m controller
- struct [sSensitivityInfo](#)
Structure to hold info about the sensitivity settings of one sensor patch.
- struct [sSensorInfo](#)
A data structure describing the sensor info about the remote DSACON32m controller.
- struct [sTactileSensorFrame](#)
A data structure describing a full tactile sensor frame read from the remote DSACON32m controller.

Public Types

- enum [eDSAEErrorCode](#) {
[E_SUCCESS](#), [E_NOT_AVAILABLE](#), [E_NO_SENSOR](#), [E_NOT_INITIALIZED](#),
[E_ALREADY_RUNNING](#), [E_FEATURE_NOT_SUPPORTED](#), [E_INCONSISTENT_DATA](#), [E_TIMEOUT](#),
[E_READ_ERROR](#), [E_WRITE_ERROR](#), [E_INSUFFICIENT_RESOURCES](#), [E_CHECKSUM_ERROR](#),
[E_CMD_NOT_ENOUGH_PARAMS](#), [E_CMD_UNKNOWN](#), [E_CMD_FORMAT_ERROR](#), [E_ACCESS_DENIED](#),
[E_ALREADY_OPEN](#), [E_CMD_FAILED](#), [E_CMD_ABORTED](#), [E_INVALID_HANDLE](#),
[E_DEVICE_NOT_FOUND](#), [E_DEVICE_NOT_OPENED](#), [E_IO_ERROR](#), [E_INVALID_PARAMETER](#),
[E_INDEX_OUT_OF_BOUNDS](#), [E_CMD_PENDING](#), [E_OVERRUN](#), [E_RANGE_ERROR](#) }
error codes returned by the remote DSACON32m tactile sensor controller

- typedef [UInt16](#) [tTexel](#)
data type for a single 'texel' (tactile sensor element)

Public Member Functions

- struct [SDH::cDSA::sControllerInfo](#) [SDH__attribute__ \(\(__packed__\)\)](#)
- struct [SDH::cDSA::sSensorInfo](#) [SDH__attribute__ \(\(__packed__\)\)](#)
- struct [SDH::cDSA::sMatrixInfo](#) [SDH__attribute__ \(\(__packed__\)\)](#)
- struct [SDH::cDSA::sSensitivityInfo](#) [SDH__attribute__ \(\(__packed__\)\)](#)
- [cDSA](#) (int debug_level=0, int port=1, char const *device_format_string="/dev/ttyS%d")
- [cDSA](#) (int debug_level, char const *_tcp_adr, int _tcp_port=13000, double _timeout=1.0)
- [~cDSA](#) ()
Destructor: clean up and delete dynamically allocated memory.
- [sControllerInfo](#) const & [GetControllerInfo](#) (void) const
Return a reference to the [sControllerInfo](#) read from the remote DSACON32m controller.
- [sSensorInfo](#) const & [GetSensorInfo](#) (void) const
Return a reference to the [sSensorInfo](#) read from the remote DSACON32m controller.
- [sMatrixInfo](#) const & [GetMatrixInfo](#) (int m) const
Return a reference to the [sMatrixInfo](#) of matrix m read from the remote DSACON32m controller.
- [sTactileSensorFrame](#) const & [GetFrame](#) () const
return a reference to the latest tactile sensor frame without reading from remote DSACON32m
- [sTactileSensorFrame](#) const & [UpdateFrame](#) ()
read the tactile sensor frame from remote DSACON32m and return a reference to it. A command to query the frame (periodically) must have been sent before.
- void [Open](#) (void) throw (cDSAException*, cSDHErrorCommunication*)
(Re-)open connection to DSACON32m controller; this is called by the constructor automatically, but is still usefull to call after a call to [Close\(\)](#)
- void [Close](#) (void) throw (cDSAException*, cSDHErrorCommunication*)
Set the framerate of the remote DSACON32m controller to 0 and close connection to it.
- void [SetFramerate](#) (UInt16 framerate, bool do_RLE=true, bool do_data_acquisition=true) throw (cDSAException*, cSDHErrorCommunication*)
- void [SetFramerateRetries](#) (UInt16 framerate, bool do_RLE=true, bool do_data_acquisition=true, unsigned int retries=0, bool ignore_exceptions=false) throw (cDSAException*, cSDHErrorCommunication*)

- [sSensitivityInfo GetMatrixSensitivity](#) (int matrix_no) throw (cDSAException*, cSDHErrorCommunication*)
- void [SetMatrixSensitivity](#) (int matrix_no, double sensitivity, bool do_all_matrices=false, bool do_reset=false, bool do_persistent=false) throw (cDSAException*, cSDHErrorCommunication*)
- void [SetMatrixThreshold](#) (int matrix_no, [UInt16](#) threshold, bool do_all_matrices=false, bool do_reset=false, bool do_persistent=false) throw (cDSAException*, cSDHErrorCommunication*)
- [UInt16 GetMatrixThreshold](#) (int matrix_no) throw (cDSAException*, cSDHErrorCommunication*)
- [tTexel GetTexel](#) (int m, int x, int y) const
- int [GetMatrixIndex](#) (int fi, int part)
- unsigned long [GetAgeOfFrame](#) ([sTactileSensorFrame](#) *frame_p)
- double [GetContactArea](#) (int m)
- [sContactInfo GetContactInfo](#) (int m)

Static Public Member Functions

- static char const * [ErrorCodeToString](#) (eDSAEErrorCode error_code)
- static char const * [ErrorCodeToString](#) ([UInt16](#) error_code)

Public Attributes

- struct VCC_EXPORT [SDH::cDSA::sTactileSensorFrame](#) [SDH__attribute__](#)
- bool [m_read_another](#)

Protected Member Functions

- struct [SDH::cDSA::sResponse](#) [SDH__attribute__](#) ((__packed__))
- void [WriteCommandWithPayload](#) ([UInt8](#) command, [UInt8](#) *payload, [UInt16](#) payload_len) throw (cDSAException*, cSDHErrorCommunication*)
- void [WriteCommand](#) ([UInt8](#) command)
- void [ReadResponse](#) ([sResponse](#) *response, [UInt8](#) command_id) throw (cDSAException*, cSDHErrorCommunication*)
- void [ReadControllerInfo](#) ([sControllerInfo](#) *_controller_info) throw (cDSAException*, cSDHErrorCommunication*)
- void [ReadSensorInfo](#) ([sSensorInfo](#) *_sensor_info) throw (cDSAException*, cSDHErrorCommunication*)
- void [ReadMatrixInfo](#) ([sMatrixInfo](#) *_matrix_info) throw (cDSAException*, cSDHErrorCommunication*)
- void [ReadFrame](#) ([sTactileSensorFrame](#) *frame_p) throw (cDSAException*, cSDHErrorCommunication*)
- void [QueryControllerInfo](#) ([sControllerInfo](#) *_controller_info) throw (cDSAException*, cSDHErrorCommunication*)
- void [QuerySensorInfo](#) ([sSensorInfo](#) *_sensor_info) throw (cDSAException*, cSDHErrorCommunication*)
- void [QueryMatrixInfo](#) ([sMatrixInfo](#) *_matrix_info, int matrix_no) throw (cDSAException*, cSDHErrorCommunication*)
- void [QueryMatrixInfos](#) (void) throw (cDSAException*, cSDHErrorCommunication*)
- void [ParseFrame](#) ([sResponse](#) *response, [sTactileSensorFrame](#) *frame_p) throw (cDSAException*)

Protected Attributes

- [cDBG dbg](#)
A stream object to print coloured debug messages.
- [cSerialBase * com](#)
the communication interface to use
- [bool do_RLE](#)
flag, true if data should be sent Run Length Encoded by the remote DSACON32m controller
- [sControllerInfo controller_info](#)
the controller info read from the remote DSACON32m controller
- [sSensorInfo sensor_info](#)
the sensor info read from the remote DSACON32m controller
- [sMatrixInfo * matrix_info](#)
the matrix infos read from the remote DSACON32m controller
- [sTactileSensorFrame frame](#)
the latest frame received from the remote DSACON32m controller
- [int nb_cells](#)
The total number of sensor cells.
- [int * texel_offset](#)
an array with the offsets of the first texel of all matrices into the whole frame
- [long read_timeout_us](#)
default timeout used for reading in us
- [cSimpleTime start_pc](#)
- [UInt32 start_dsa](#)
- [tTexel contact_area_cell_threshold](#)
threshold of texel cell value for detecting contacts with `GetContactArea`
- [tTexel contact_force_cell_threshold](#)
threshold of texel cell value for detecting forces with `GetContactForce`
- [double force_factor](#)
additional calibration factor for forces in `GetContactForce`
- [double calib_pressure](#)
- [double calib_voltage](#)
see `calib_pressure`

Friends

- VCC_EXPORT std::ostream & [operator<<](#) (std::ostream &stream, [cDSA::sResponse](#) const &response)

10.11.1 Detailed Description

[SDH::cDSA](#) is the end user interface class to access the DSA32m, the tactile sensor controller of the SDH. Class to communicate with the tactile sensor controller DSA32m of the SDH

Bug

[cDSAException](#): Checksum Error on Windows console While communicating with the tactile sensor controller a "cDSAException: Checksum Error" might be thrown once in a while. This seems to happen only when the program is started from a native windows command console (cmd.exe), but not e.g. when the program is started from a cygwin console. Strange. **Workaround** is to catch the exception and ignore the frame.

=> **Resolved in SDHLibrary-C++ v0.0.2.1**

Problem was an overrun of the windows input buffer, e.g. on heavy processor load.

10.11.2 Member Typedef Documentation

10.11.2.1 typedef UInt16 SDH::cDSA::tTexel

data type for a single 'texel' (tactile sensor element)

10.11.3 Member Enumeration Documentation

10.11.3.1 enum SDH::cDSA::eDSAEErrorCode

error codes returned by the remote DSA32m tactile sensor controller

Enumerator:

E_SUCCESS

E_NOT_AVAILABLE

E_NO_SENSOR

E_NOT_INITIALIZED

E_ALREADY_RUNNING

E_FEATURE_NOT_SUPPORTED

E_INCONSISTENT_DATA

E_TIMEOUT

E_READ_ERROR

E_WRITE_ERROR
E_INSUFFICIENT_RESOURCES
E_CHECKSUM_ERROR
E_CMD_NOT_ENOUGH_PARAMS
E_CMD_UNKNOWN
E_CMD_FORMAT_ERROR
E_ACCESS_DENIED
E_ALREADY_OPEN
E_CMD_FAILED
E_CMD_ABORTED
E_INVALID_HANDLE
E_DEVICE_NOT_FOUND
E_DEVICE_NOT_OPENED
E_IO_ERROR
E_INVALID_PARAMETER
E_INDEX_OUT_OF_BOUNDS
E_CMD_PENDING
E_OVERRUN
E_RANGE_ERROR

10.11.4 Constructor & Destructor Documentation

10.11.4.1 `cDSA::cDSA (int debug_level = 0, int port = 1, char const * device_format_string =
"/dev/ttyS%d")`

Constructor for `cDSA`. This constructs a `cDSA` object to communicate with the remote DSA32m controller within the SDH via RS232.

The connection is opened and established, and the sensor, controller and matrix info is queried from the remote DSA32m controller. This initialization may take up to 9 seconds, since the DSA32m controller needs > 8 seconds for "booting". If the SDH is already powered for some time then this will be much quicker.

Parameters

<i>debug_level</i>	- the level of debug messages to be printed: <ul style="list-style-type: none">• if > 0 (1,2,3...) then debug messages of <code>cDSA</code> itself are printed• if > 1 (2,3,...) then debug messages of the low level communication interface object are printed too
<i>port</i>	- the RS232 port to use for communication. (port 0 = ttyS0 = COM1, port 1 = ttyS1 = COM2, ...)
<i>device_format_string</i>	- a format string (C string) for generating the device name, like "/dev/ttyS%d" (default) or "/dev/ttyUSB%d". Must contain a d where the port number should be inserted. This char array is duplicated on construction.
	When compiled with VGG (MS Visual Studio) this is not used

10.11.4.2 `cDSA::cDSA (int debug_level, char const * _tcp_adr, int _tcp_port = 13000, double _timeout = 1.0)`

Constructor for [cDSA](#). This constructs a [cDSA](#) object to communicate with the remote DSACON32m controller within the SDH via TCP/IP (ethernet).

The connection is opened and established, and the sensor, controller and matrix info is queried from the remote DSACON32m controller. This initialization may take up to 9 seconds, since the DSACON32m controller needs > 8 seconds for "booting". If the SDH is already powered for some time then this will be much quicker.

Parameters

<i>debug_level</i>	- the level of debug messages to be printed: <ul style="list-style-type: none"> • if > 0 (1,2,3,...) then debug messages of cDSA itself are printed • if > 1 (2,3,...) then debug messages of the low level communication interface object are printed too
<i>_tcp_adr</i>	: The tcp host address of the SDH . Either a numeric IP as string or a host-name
<i>_tcp_port</i>	: the tcp port number on the SDH to connect to
<i>_timeout</i>	: The timeout to use: <ul style="list-style-type: none"> • <= 0 : wait forever (default) • T : wait for T seconds

10.11.4.3 `cDSA::~~cDSA ()`

Destructur: clean up and delete dynamically allocated memory.

10.11.5 Member Function Documentation

10.11.5.1 `void cDSA::Close (void) throw (cDSAException*, cSDHErrorCommunication*)`

Set the framerate of the remote DSACON32m controller to 0 and close connection to it.

10.11.5.2 `char const * cDSA::ErrorCodeToString (eDSAErrorCode error_code)`
[static]

Return a short string description for *error_code*

Parameters

<i>error_code</i>	- error code as returned from the remote DSACON32m tactile sensor controller
-------------------	--

Returns

short string description for *error_code*

10.11.5.3 `static char const* SDH::cDSA::ErrorCodeToString (UInt16 error_code)`
`[inline, static]`

10.11.5.4 `unsigned long SDH::cDSA::GetAgeOfFrame (sTactileSensorFrame * frame_p)`
`[inline]`

return age of frame in ms (time in ms from frame sampling until now)

10.11.5.5 `double cDSA::GetContactArea (int m)`

Return contact area in mm*mm for matrix with index *m*

10.11.5.6 `cDSA::sContactInfo cDSA::GetContactInfo (int m)`

Return a [sContactInfo](#) struct (force,cog_x,cog_y,area) of contact force and center of gravity and contact area of that force for matrix with index *m*

10.11.5.7 `sControllerInfo const& SDH::cDSA::GetControllerInfo (void) const`
`[inline]`

Return a reference to the [sControllerInfo](#) read from the remote DSACON32m controller.

10.11.5.8 `sTactileSensorFrame const& SDH::cDSA::GetFrame () const` `[inline]`

return a reference to the latest tactile sensor frame without reading from remote DSACON32m

10.11.5.9 `int SDH::cDSA::GetMatrixIndex (int fi, int part)` `[inline]`

return the matrix index of the sensor matrix attached to finger with index *fi* [1..3] and *part* [0,1] = [proximal,distal]

10.11.5.10 `sMatrixInfo const& SDH::cDSA::GetMatrixInfo (int m) const` `[inline]`

Return a reference to the [sMatrixInfo](#) of matrix *m* read from the remote DSACON32m controller.

10.11.5.11 `cDSA::sSensitivityInfo cDSA::GetMatrixSensitivity (int matrix_no) throw (cDsaException*, cSDHErrorCommunication*)`

Send command to get matrix sensitivity. Returns sensitivities of matrix no *matrix_no*.

A struct is returned containing the members *error_code* - see DSACON32 Command Set Reference Manual *adj_flags* - see DSACON32 Command Set Reference Manual *cur_sens* - the currently set sensitivity as float [0.0 .. 1.0] (0.0 is minimum, 1.0 is maximum sensitivity) *fact_sens* - the factory sensitivity as float [0.0 .. 1.0] (0.0 is minimum, 1.0 is maximum sensitivity)

Raises a [cDsaException](#) in case of invalid responses from the remote DSACON32m controller.

Bug

With DSACON32m firmware R218 and before this did not work, instead the factory default (0.5) was always reported

=> **Resolved in DSACON32m firmware R268**

10.11.5.12 `UInt16 cDSA::GetMatrixThreshold (int matrix_no) throw (cDsaException*, cSDHErrorCommunication*)`

Send command to get matrix threshold. Returns threshold of matrix no *matrix_no*.

Returns

threshold - the currently set threshold as integer [0 .. 4095] (0 is minimum, 4095 is maximum threshold)

Raises a [cDsaException](#) in case of invalid responses from the remote DSACON32m controller.

Remarks

Getting the matrix threshold is only possible if the DSACON32m firmware is R268 or above.

10.11.5.13 `sSensorInfo const& SDH::cDSA::GetSensorInfo (void) const [inline]`

Return a reference to the [sSensorInfo](#) read from the remote DSACON32m controller.

10.11.5.14 `cDSA::tTexel cDSA::GetTexel (int m, int x, int y) const`

Return texel value at column *x* row *y* of matrix *m* of the last frame

10.11.5.15 `void cDSA::Open (void) throw (cDSAException*, cSDHErrorCommunication*)`

(Re-)open connection to DSA32m controller, this is called by the constructor automatically, but is still usefull to call after a call to [Close\(\)](#)

10.11.5.16 `void cDSA::ParseFrame (sResponse * response, sTactileSensorFrame * frame_p) throw (cDSAException*)` [protected]

Parse a full frame response from remote DSA

10.11.5.17 `void cDSA::QueryControllerInfo (sControllerInfo * _controller_info) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

Send command to request controller info from remote DSA32m. Read and parse the response from the remote DSA32m.

10.11.5.18 `void cDSA::QueryMatrixInfo (sMatrixInfo * _matrix_info, int matrix_no) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

Send command to request matrix info from remote DSA32m. Read and parse the response from the remote DSA32m.

10.11.5.19 `void cDSA::QueryMatrixInfos (void) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

Query all matrix infos

10.11.5.20 `void cDSA::QuerySensorInfo (sSensorInfo * _sensor_info) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

Send command to request sensor info from remote DSA32m. Read and parse the response from the remote DSA32m.

10.11.5.21 `void cDSA::ReadControllerInfo (sControllerInfo * _controller_info) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

Read and parse a controller info response from the remote DSA

10.11.5.22 `void cDSA::ReadFrame (sTactileSensorFrame * frame_p) throw (cDSAException*, cSDHErrorCommunication*)` [protected]

read and parse a full frame response from remote DSA

10.11.5.23 void cDSA::ReadMatrixInfo (sMatrixInfo * *_matrix_info*) throw
(cDsaException*, cSDHErrorCommunication*) [protected]

Read and parse a matrix info response from the remote DSA

10.11.5.24 void cDSA::ReadResponse (sResponse * *response*, UInt8 *command_id*) throw
(cDsaException*, cSDHErrorCommunication*) [protected]

Read any response from the remote DSA CON32m, expect the *command_id* as command id

- tries to find the preamble within the next at most DSA_MAX_PREAMBLE_SEARCH bytes from the device
- reads the packet id and size
- reads all data indicated by the size read
- if there is enough space in the payload of the response then the received data is stored there if there is not enough space in the payload of the response then the data is received but forgotten (to keep the communication line clear)
- if sent, the CRC checksum is read and the data is checked. For invalid data an exception is thrown

10.11.5.25 void cDSA::ReadSensorInfo (sSensorInfo * *_sensor_info*) throw
(cDsaException*, cSDHErrorCommunication*) [protected]

Read and parse a sensor info response from the remote DSA

- 10.11.5.26 `struct SDH::cDSA::sResponse SDH::cDSA::SDH__attribute__ (`
`(__packed__)) [protected]`
- 10.11.5.27 `struct SDH::cDSA::sControllerInfo SDH::cDSA::SDH__attribute__ (`
`(__packed__))`
- 10.11.5.28 `struct SDH::cDSA::sSensorInfo SDH::cDSA::SDH__attribute__ (`
`(__packed__))`
- 10.11.5.29 `struct SDH::cDSA::sSensitivityInfo SDH::cDSA::SDH__attribute__ (`
`(__packed__))`
- 10.11.5.30 `struct SDH::cDSA::sMatrixInfo SDH::cDSA::SDH__attribute__ (`
`(__packed__))`
- 10.11.5.31 `void cDSA::SetFramerate (UInt16 framerate, bool do_RLE = true,`
`bool do_data_acquisition = true) throw (cDSAException*,`
`cSDHErrorCommunication*)`

Set the *framerate* for querying full frames.

Parameters

<i>framerate</i>	- rate of frames.
<i>do_RLE</i>	- flag, if true then use RLE (run length encoding) for sending frames
<i>do_data_acquisition</i>	- flag, enable or disable data acquisition. Must be true if you want to get new frames

- Use *framerate=0* and *do_data_acquisition=false* to make the remote DSACON32m in SDH stop sending frames
- Use *framerate=0* and *do_data_acquisition=true* to read a single frame
- Use *framerate>0* and *do_data_acquisition=true* to make the remote DSACON32m in SDH send frames at the highest possible rate (for now: 30 FPS (frames per second)).

Bug

SCHUNK-internal bugzilla ID: Bug 680
 With DSACON32m firmware R276 and after and SDHLibrary-C++ v0.0.1.15 and before stopping of the sending did not work.
=> Resolved in SDHLibrary-C++ v0.0.1.16

Bug

SCHUNK-internal bugzilla ID: Bug 680
 With DSACON32m firmware before R276 and SDHLibrary-C++ v0.0.1.16 and before acquiring of single frames did not work
=> Resolved in SDHLibrary-C++ v0.0.1.17

Bug

SCHUNK-internal bugzilla ID: Bug 703

With DSACON32m firmware R288 and before and SDHLibrary-C++ v0.0.2.1 and before tactile sensor frames could not be read reliably in single frame mode.

=> **Resolved in DSACON32m firmware 2.9.0.0**

=> **Resolved in SDHLibrary-C++ v0.0.2.1 with workaround for older DSACON32m firmwares**

10.11.5.32 `void cDSA::SetFramerateRetries (UInt16 framerate, bool do_RLE = true, bool do_data_acquisition = true, unsigned int retries = 0, bool ignore_exceptions = false) throw (cDSAException*, cSDHErrorCommunication*)`

Set the *framerate* for querying full frames.

Parameters

<i>framerate</i>	- rate of frames. 0 will make the remote DSACON32m in SDH stop sending frames, any value > 0 will make the remote DSACON32m in SDH send frames at the highest possible rate (for now: 30 FPS (frames per second)).
<i>do_RLE</i>	- flag, if true then use RLE (run length encoding) for sending frames
<i>do_data_acquisition</i>	- flag, enable or disable data acquisition. Must be true if you want to get new frames
<i>retries</i>	- number of times the sending will be retried in case of an error (like timeout while waiting for response)
<i>ignore_exceptions</i>	- flag, if true then exceptions are ignored in case of error. After <i>retries</i> tries the function just returns even in case of an error

10.11.5.33 `void cDSA::SetMatrixSensitivity (int matrix_no, double sensitivity, bool do_all_matrices = false, bool do_reset = false, bool do_persistent = false) throw (cDSAException*, cSDHErrorCommunication*)`

Send command to set matrix sensitivity to *sensitivity* as float [0.0 .. 1.0] (0.0 is minimum, 1.0 is maximum sensitivity). If *do_all_matrices* is true then the *sensitivity* is set for all matrices. If *do_reset* is true then the sensitivity is reset to the factory default. If *do_persistent* is true then the sensitivity is saved persistently to configuration memory and will thus remain after the next power off/power on cycle and will become the new factory default value. If *do_persistent* is false (default) then the value will be reset to default on the next power off/power on cycle

Warning

PLEASE NOTE: the maximum write endurance of the configuration memory is about 100.000 times!

Raises a [cDSAException](#) in case of invalid responses from the remote DSACON32m controller.

Remarks

Setting the matrix sensitivity persistently is only possible if the DSA32m firmware is R268 or above.

```
10.11.5.34 void cDSA::SetMatrixThreshold ( int matrix_no, UInt16 threshold, bool
      do_all_matrices = false, bool do_reset = false, bool do_persistent = false )
      throw (cDSAException*, cSDHErrorCommunication*)
```

Send command to set matrix threshold to *threshold* as integer [0 .. 4095] (0 is minimum, 4095 is maximum threshold). If *do_all_matrices* is true then the *threshold* is set for all matrices. If *do_reset* is true then the threshold is reset to the factory default. If *do_persistent* is true then the threshold is saved persistently to configuration memory and will thus remain after the next power off/power on cycle and will become the new factory default value. If *do_persistent* is false (default) then the value will be reset to default on the next power off/power on cycle

Warning

PLEASE NOTE: the maximum write endurance of the configuration memory is about 100.000 times!

Raises a [cDSAException](#) in case of invalid responses from the remote DSA32m controller.

Remarks

Getting the matrix threshold is only possible if the DSA32m firmware is R268 or above.

```
10.11.5.35 sTactileSensorFrame const& SDH::cDSA::UpdateFrame ( ) [inline]
```

read the tactile sensor frame from remote DSA32m and return a reference to it. A command to query the frame (periodically) must have been sent before.

10.11.5.36 void SDH::cDSA::WriteCommand (UInt8 *command*) [inline, protected]

10.11.5.37 void cDSA::WriteCommandWithPayload (UInt8 *command*, UInt8 * *payload*, UInt16 *payload_len*) throw (cDSAException*, cSDHErrorCommunication*) [protected]

10.11.6 Friends And Related Function Documentation

10.11.6.1 VCC_EXPORT std::ostream& operator<< (std::ostream & *stream*, cDSA::sResponse const & *response*) [friend]

10.11.7 Member Data Documentation

10.11.7.1 double SDH::cDSA::calib_pressure [protected]

For the voltage to pressure conversion in _VoltageToPressure() enter one pressure/voltage measurement here (from demo-dsa.py --calibration):

10.11.7.2 double SDH::cDSA::calib_voltage [protected]

see calib_pressure

10.11.7.3 cSerialBase* SDH::cDSA::com [protected]

the communication interface to use

10.11.7.4 tTexel SDH::cDSA::contact_area_cell_threshold [protected]

threshold of texel cell value for detecting contacts with GetContactArea

10.11.7.5 tTexel SDH::cDSA::contact_force_cell_threshold [protected]

threshold of texel cell value for detecting forces with GetContactForce

10.11.7.6 sControllerInfo SDH::cDSA::controller_info [protected]

the controller info read from the remote DSACON32m controller

10.11.7.7 cDBG SDH::cDSA::dbg [protected]

A stream object to print coloured debug messages.

10.11.7.8 bool SDH::cDSA::do_RLE [protected]

flag, true if data should be sent Run Length Encoded by the remote DSACON32m controller

10.11.7.9 double SDH::cDSA::force_factor [protected]

additional calibration factor for forces in GetContactForce

10.11.7.10 sTactileSensorFrame SDH::cDSA::frame [protected]

the latest frame received from the remote DSACON32m controller

10.11.7.11 bool SDH::cDSA::m_read_another

flag, if True then the [ReadFrame\(\)](#) function will read tactile sensor frames until a time-out occurs. This will ignore intermediate frames as long as new ones are available. This was usefull some time in the past, or if you have a slow computer that cannot handle incoming frames fast enough. If False then any completely read valid frame will be reported. With new computers and fast communication like via TCP this should remain at "False".

10.11.7.12 sMatrixInfo* SDH::cDSA::matrix_info [protected]

the matrix infos read from the remote DSACON32m controller

10.11.7.13 int SDH::cDSA::nb_cells [protected]

The total number of sensor cells.

10.11.7.14 long SDH::cDSA::read_timeout_us [protected]

default timeout used for reading in us

**10.11.7.15 struct VCC_EXPORT SDH::cDSA::sTactileSensorFrame
SDH::cDSA::SDH__attribute__****10.11.7.16 sSensorInfo SDH::cDSA::sensor_info** [protected]

the sensor info read from the remote DSACON32m controller

10.11.7.17 **UInt32** SDH::cDSA::start_dsa [protected]

10.11.7.18 **cSimpleTime** SDH::cDSA::start_pc [protected]

10.11.7.19 **int*** SDH::cDSA::texel_offset [protected]

an array with the offsets of the first texel of all matrices into the whole frame

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

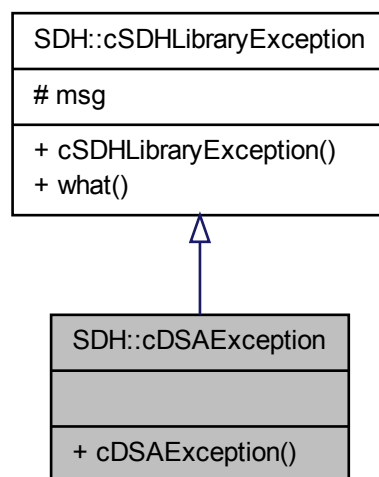
- [sdh/dsa.h](#)
- [sdh/dsa.cpp](#)

10.12 SDH::cDSAException Class Reference

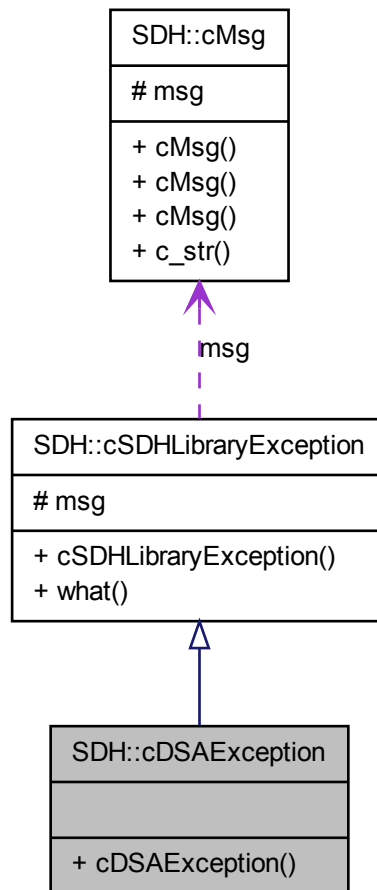
Derived exception class for low-level DSA related exceptions.

```
#include <dsa.h>
```

Inheritance diagram for SDH::cDSAException:



Collaboration diagram for SDH::cDSAException:



Public Member Functions

- [`cDSAException`](#) ([`cMsg`](#) const &`_msg`)

10.12.1 Detailed Description

Derived exception class for low-level DSA related exceptions.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 SDH::cDSAException::cDSAException (cMsg const & _msg) [inline]

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

- [sdh/dsa.h](#)

10.13 cDSAUpdater Class Reference

Class to create an updater thread for continuously updating tactile sensor data.

```
#include <dsaboost.h>
```

Public Member Functions

- [cDSAUpdater](#) (cDSA * _ts, int _error_threshold=DEFAULT_ERROR_THRESHOLD)
- void [interrupt](#) ()
interrupt the updater thread

Static Public Attributes

- static int const [DEFAULT_ERROR_THRESHOLD](#) = 16
default error threshold, see parameter error_threshold in CTOR

10.13.1 Detailed Description

Class to create an updater thread for continuously updating tactile sensor data. Uses boost::thread from the boost library (<http://www.boost.org>)!

10.13.2 Constructor & Destructor Documentation

10.13.2.1 cDSAUpdater::cDSAUpdater (cDSA * _ts, int _error_threshold = DEFAULT_ERROR_THRESHOLD)

CTOR: start an updater thread for the connected tactile sensor _ts

Make remote DSA send frames. Create a thread that updates _ts->frame continuously

Parameters

<code>_ts</code>	- ptr to already initialized cDSA tactile sensor object
------------------	---

<i><code>_error_threshold</code></i>	<p>- the number of errors that causes a "reset" of the connection to the remote DSA: communication errors with the remote DSA32m controllers are counted,</p> <ul style="list-style-type: none"> • every error increments the error level, • every correct communication decrements the error level (down to 0) if the error level rises above <code>ERROR_THRESHOLD</code> then the connection is closed and the reopened. This is usefull to recover from emergency stop since the hands loose power temporarily in that case.
--------------------------------------	--

10.13.3 Member Function Documentation

10.13.3.1 `void cDSAUpdater::interrupt () [inline]`

interrupt the updater thread

10.13.4 Member Data Documentation

10.13.4.1 `int const cDSAUpdater::DEFAULT_ERROR_THRESHOLD = 16 [static]`

default error threshold, see parameter *error_threshold* in CTOR

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

- [demo/dsaboost.h](#)
- [demo/dsaboost.cpp](#)

10.14 SDH::cHexByteString Class Reference

dummy class for (debug) stream output of bytes as list of hex values

```
#include <dbg.h>
```

Public Member Functions

- [cHexByteString](#) (char const *_bytes, int _len)
ctor: create a [cHexByteString](#) with _len bytes at _bytes

Friends

- `VCC_EXPORT std::ostream & operator<< (std::ostream &stream, cHexByteString const &s)`

output the bytes in s to stream as a list of space separated hex bytes (without 0x prefix)

10.14.1 Detailed Description

dummy class for (debug) stream output of bytes as list of hex values

10.14.2 Constructor & Destructor Documentation

10.14.2.1 SDH::cHexString::cHexString (char const * *_bytes*, int *_len*) [inline]

ctor: create a [cHexString](#) with *_len* bytes at *_bytes*

10.14.3 Friends And Related Function Documentation

10.14.3.1 VCC_EXPORT std::ostream& operator<< (std::ostream & *stream*,
cHexString const & *s*) [friend]

output the bytes in *s* to *stream* as a list of space separated hex bytes (without 0x prefix)

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

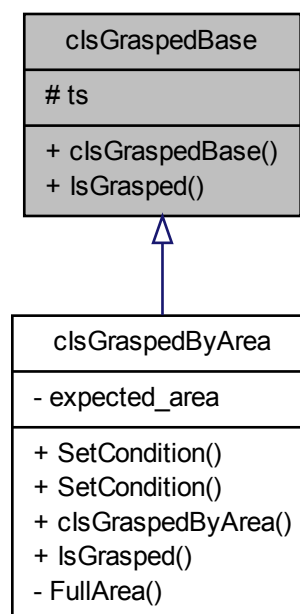
- [sdh/dbg.h](#)

10.15 clsGraspedBase Class Reference

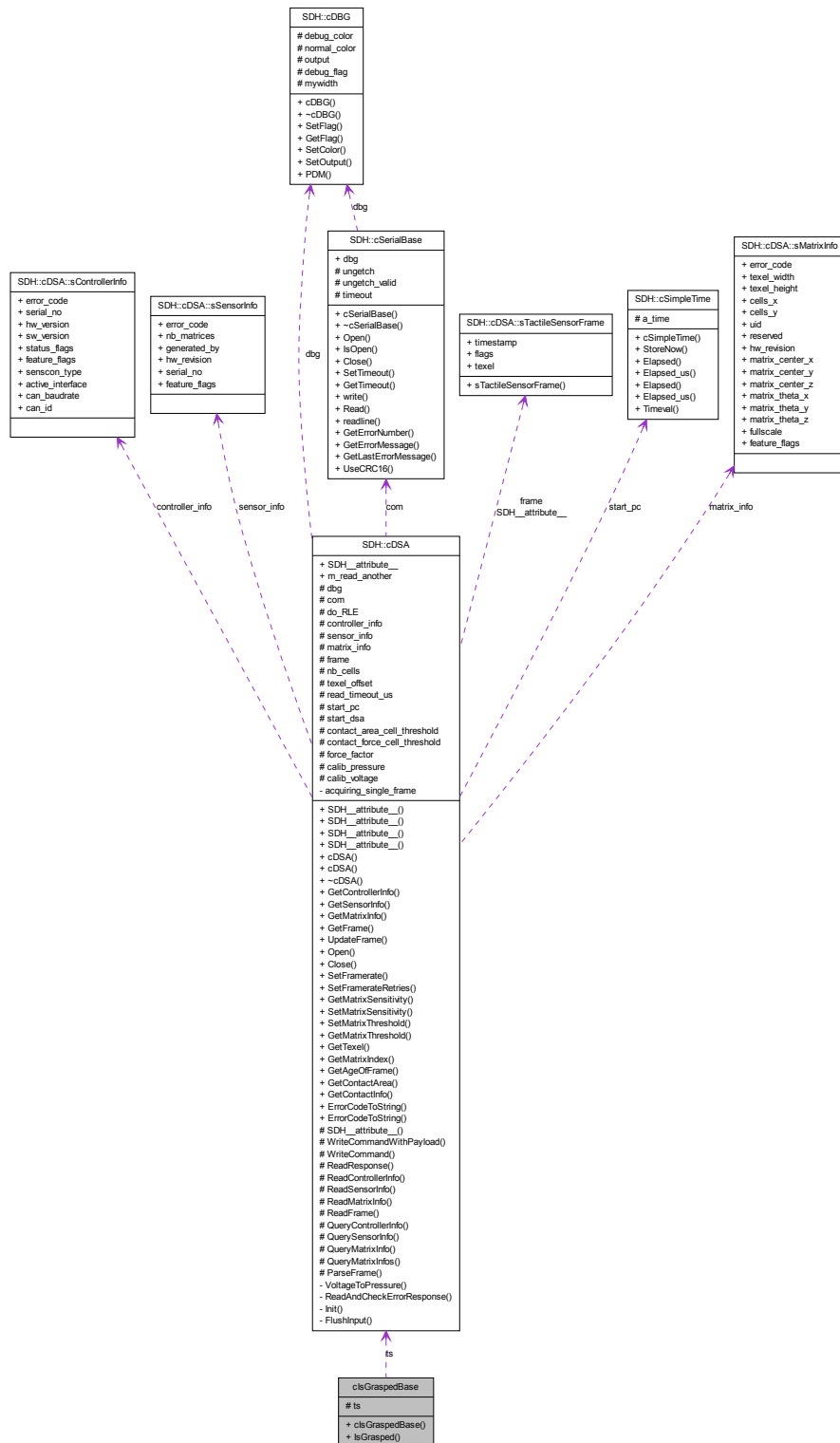
abstract base class for calculation of IsGrasped condition using tactile sensor information

```
#include <dsaboost.h>
```

Inheritance diagram for `clsGraspedBase`:



Collaboration diagram for cIsGraspedBase:



Public Member Functions

- [cIsGraspedBase](#) ([SDH::cDSA](#) *_ts)
- virtual bool [IsGrasped](#) (void)=0

Protected Attributes

- [SDH::cDSA](#) * ts
ptr to the cDSA tactile sensor object to use

10.15.1 Detailed Description

abstract base class for calculation of IsGrasped condition using tactile sensor information

10.15.2 Constructor & Destructor Documentation

10.15.2.1 [clsGraspedBase::clsGraspedBase](#) ([SDH::cDSA](#) * [_ts](#)) [inline]

10.15.3 Member Function Documentation

10.15.3.1 virtual bool [clsGraspedBase::IsGrasped](#) (void) [pure virtual]

Implemented in [cIsGraspedByArea](#).

10.15.4 Member Data Documentation

10.15.4.1 [SDH::cDSA](#)* [cIsGraspedBase::ts](#) [protected]

ptr to the cDSA tactile sensor object to use

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

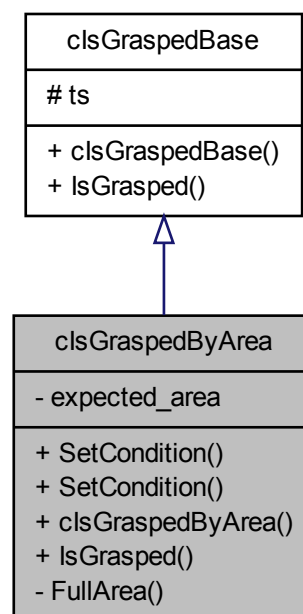
- demo/[dsaboost.h](#)

10.16 cIsGraspedByArea Class Reference

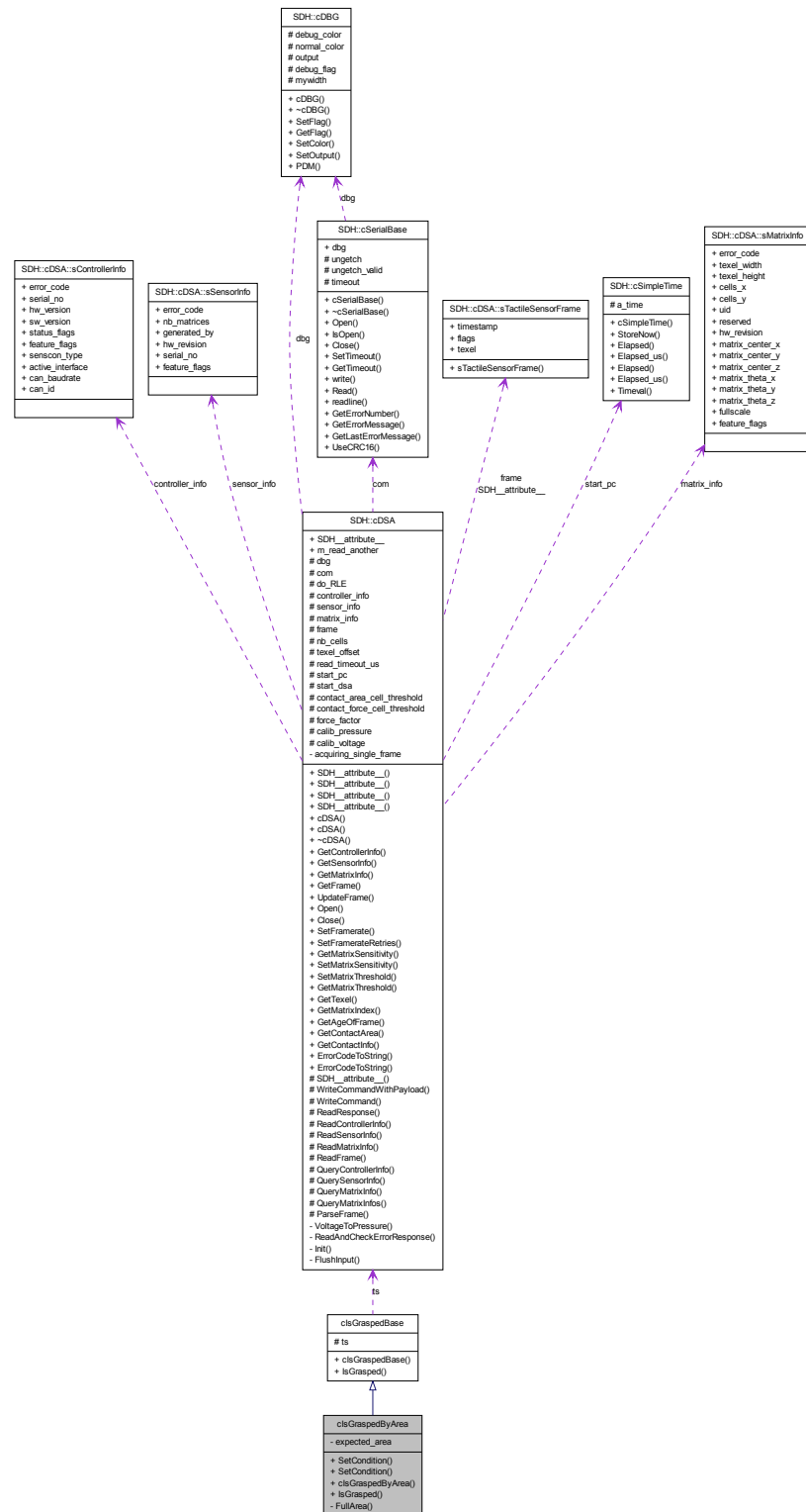
class for calculation of IsGrasped condition using an expected area of contact measured by the tactile sensors

```
#include <dsaboost.h>
```

Inheritance diagram for clsGraspedByArea:



Collaboration diagram for cIsGraspedByArea:



Public Member Functions

- void [SetCondition](#) (double eaf0p, double eaf0d, double eaf1p, double eaf1d, double eaf2p, double eaf2d)
- void [SetCondition](#) (double *eafx)
overloaded variant which uses an array of doubles instead of 6 single double parameters
- [cIsGraspedByArea](#) (SDH::cDSA *_ts)
default constructor which initializes the internal date
- virtual bool [IsGrasped](#) (void)

10.16.1 Detailed Description

class for calculation of IsGrasped condition using an expected area of contact measured by the tactile sensors

10.16.2 Constructor & Destructor Documentation

10.16.2.1 [cIsGraspedByArea::cIsGraspedByArea](#) (SDH::cDSA * _ts)

default constructor which initializes the internal date

10.16.3 Member Function Documentation

10.16.3.1 [bool cIsGraspedByArea::IsGrasped](#) (void) `[virtual]`

Implementation of is grasped condition.

Returns

true if the object is grasped according to the actual tactile sensor data in *ts* and the condition set with [SetCondition\(\)](#)

Implements [cIsGraspedBase](#).

10.16.3.2 [void cIsGraspedByArea::SetCondition](#) (double eaf0p, double eaf0d, double eaf1p, double eaf1d, double eaf2p, double eaf2d)

set the is grasped condition

Parameters

<i>eaf0p</i>	- expected area of finger 0 proximal for detecting grasp condition, value [0..1] with 0=no area, 1=full area
--------------	--

<i>eaf0d</i>	- expected area of finger 0 distal for detecting grasp condition, value [0..1] with 0=no area, 1=full area
<i>eaf1p</i>	- ...
<i>eaf1d</i>	- ...
<i>eaf2p</i>	- ...
<i>eaf2d</i>	- ...

10.16.3.3 void clsGraspedByArea::SetCondition (double * eafx)

overloaded variant which uses an array of doubles instead of 6 single double parameters

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

- demo/[dsaboost.h](#)
- demo/[dsaboost.cpp](#)

10.17 SDH::cMsg Class Reference

Class for short, fixed maximum length text messages.

```
#include <sdhexception.h>
```

Public Member Functions

- [cMsg](#) ()
Default constructor, init message to empty string.
- [cMsg](#) (cMsg const &other)
Copy constructor, copy message content of other object to this object.
- [cMsg](#) (char const *fmt,...) [SDH__attribute__](#)((format(printf
Constructor with printf like format, argument parameters.
- char const * [c_str](#) () const
Return the C-string representation of the messag in this object.

Protected Types

- enum { [eMAX_MSG](#) = 512 }
anonymous enum instead of define macros

Protected Attributes

- char [msg](#) [eMAX_MSG]

10.17.1 Detailed Description

Class for short, fixed maximum length text messages. Simple message objects for short, fixed maximum length text messages, but with printf like initialization.

An object of type [SDH::cMsg](#) contains an ASCII-Z string of maximum length [eMAX_MSG](#). It can be initialized with a 'printf-style' format string by the constructor. It is used in the SDHLibrary to further specify the cause of an exception in human readable form.

See [SDH::cSDHLibraryException::cSDHLibraryException\(\)](#) for exemplary use.

10.17.2 Member Enumeration Documentation

10.17.2.1 anonymous enum [protected]

anonymous enum instead of define macros

Enumerator:

eMAX_MSG maximum length in bytes of a message to store

10.17.3 Constructor & Destructor Documentation

10.17.3.1 cMsg::cMsg ()

Default constructor, init message to empty string.

10.17.3.2 cMsg::cMsg (cMsg const & other)

Copy constructor, copy message content of other object to this object.

10.17.3.3 cMsg::cMsg (char const * fmt, ...)

Constructor with printf like format, argument parameters.

10.17.4 Member Function Documentation

10.17.4.1 char const * cMsg::c_str () const

Return the C-string representation of the messag in this object.

10.17.5 Member Data Documentation

10.17.5.1 `char SDH::cMsg::msg[eMAX_MSG]` `[protected]`

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

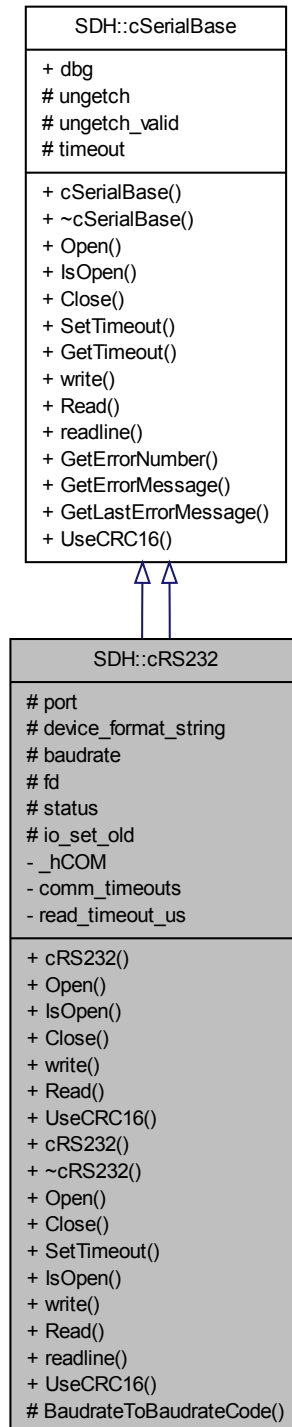
- [sdh/sdhexception.h](#)
- [sdh/sdhexception.cpp](#)

10.18 SDH::cRS232 Class Reference

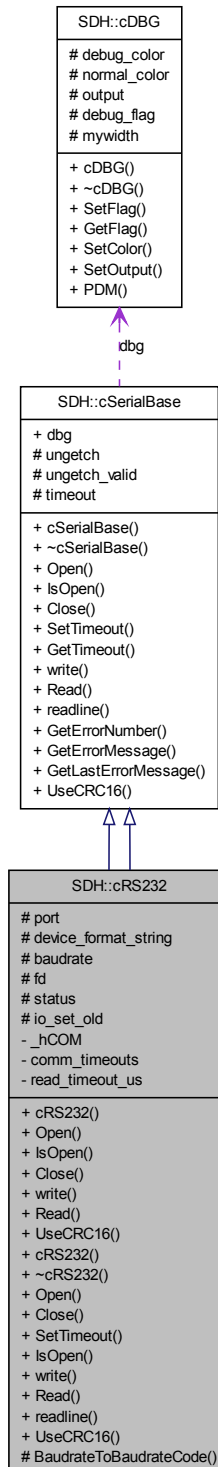
Low-level communication class to access a serial port on Cygwin and Linux.

```
#include <rs232-vcc.h>
```

Inheritance diagram for SDH::cRS232:



Collaboration diagram for SDH::cRS232:



Public Member Functions

- [cRS232](#) (int _port, unsigned long _baudrate, double _timeout, char const *_device_format_string="/dev/ttyS%d")
- void [Open](#) (void) throw (cRS232Exception*)
- bool [IsOpen](#) (void) throw ()
Return true if port to RS232 is open.
- void [Close](#) (void) throw (cRS232Exception*)
Close the previously opened rs232 port.
- int [write](#) (char const *ptr, int len=0) throw (cRS232Exception*)
Write data to a previously opened port.
- ssize_t [Read](#) (void *data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cRS232Exception*)
- virtual bool [UseCRC16](#) ()
overloaded from [cSerialBase::UseCRC16](#) since we want to use a CRC16 to protect binary RS232 communication
- [cRS232](#) (int _port, unsigned long _baudrate, double _timeout, char const *_device_format_string="")
- [~cRS232](#) (void)
- void [Open](#) (void) throw (cRS232Exception*)
Open rs232 port port.
- void [Close](#) (void) throw (cRS232Exception*)
Close the previously opened communication channel.
- virtual void [SetTimeout](#) (double _timeout) throw (cSerialBaseException*)
set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- bool [IsOpen](#) () throw ()
Return true if communication channel is open.
- int [write](#) (char const *ptr, int len=0) throw (cRS232Exception*)
Write data to a previously opened port.
- ssize_t [Read](#) (void *data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cRS232Exception*)
- char * [readline](#) (char *line, int size, char *eol, bool return_on_less_data) throw (cRS232Exception*)
- virtual bool [UseCRC16](#) ()
overloaded from [cSerialBase::UseCRC16](#) since we want to use a CRC16 to protect binary RS232 communication

Protected Member Functions

- `tcflag_t BaudrateToBaudrateCode` (unsigned long `baudrate`) throw (cRS232Exception*)

Translate a baudrate given as unsigned long into a baudrate code for struct termios.

Protected Attributes

- `int port`
the RS232 portnumber to use
- `std::string device_format_string`
the sprintf format string to generate the device name from the port, see Constructor
- `unsigned long baudrate`
the baudrate in bit/s
- `int fd`
the file descriptor of the RS232 port
- `int status`
- `termios io_set_old`

10.18.1 Detailed Description

Low-level communication class to access a serial port on Cygwin and Linux. Low-level communication class to access a serial port on VCC Windows.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 `cRS232::cRS232 (int _port, unsigned long _baudrate, double _timeout, char const * _device_format_string = "/dev/ttyS%d")`

Constructor: constructs an object to communicate with an SDH via RS232

Parameters

<code>_port</code>	- rs232 device number: 0='COM1'='/dev/ttyS0', 1='COM2'='/dev/ttyS1', ...
<code>_baudrate</code>	- the baudrate in bit/s
<code>_timeout</code>	- the timeout in seconds
<code>_device_format_string</code>	- a format string (C string) for generating the device name, like "/dev/ttyS%d" (default) or "/dev/ttyUSB%d". Must contain a d where the port number should be inserted. This char array is duplicated on construction

10.18.2.2 `SDH::cRS232::cRS232 (int _port, unsigned long _baudrate, double _timeout, char const * _device_format_string = " ")`

Constructor: constructs an object to communicate with an SDH via RS232

Parameters

<code>_port</code>	- rs232 device number: 0='COM1'='/dev/ttyS0', 1='COM2'='/dev/ttyS1', ...
<code>_baudrate</code>	- the baudrate in bit/s
<code>_timeout</code>	- the timeout in seconds
<code>_device_format_string</code>	- ignored for this VCC version

10.18.2.3 `cRS232::~~cRS232 (void)`

10.18.3 Member Function Documentation

10.18.3.1 `tcflag_t cRS232::BaudrateToBaudrateCode (unsigned long baudrate) throw (cRS232Exception*)` [protected]

Translate a baudrate given as unsigned long into a baudrate code for struct termios.

10.18.3.2 `void cRS232::Close (void) throw (cRS232Exception*)` [virtual]

Close the previously opened rs232 port.

Implements [SDH::cSerialBase](#).

10.18.3.3 `void SDH::cRS232::Close (void) throw (cRS232Exception*)` [virtual]

Close the previously opened communication channel.

Implements [SDH::cSerialBase](#).

10.18.3.4 `bool SDH::cRS232::IsOpen () throw ()` [inline, virtual]

Return true if communication channel is open.

Implements [SDH::cSerialBase](#).

10.18.3.5 `bool cRS232::IsOpen (void) throw ()` [virtual]

Return true if port to RS232 is open.

Implements [SDH::cSerialBase](#).

10.18.3.6 void SDH::cRS232::Open (void) throw (cRS232Exception*) [virtual]

Open rs232 port *port*.

Open the device with the parameters provided in the constructor

Implements [SDH::cSerialBase](#).

10.18.3.7 void cRS232::Open (void) throw (cRS232Exception*) [virtual]

Open the device as configured by the parameters given to the constructor

Bug

The binary communication introduced in firmware 0.0.2.15 and SDHLibrary-C++ 0.0.2.0 did not work properly on Linux when RS232 was used for communication. See also:

- [Bug 1011: Bug: Binary communication does not work via RS232 on Linux](#)
- [SDH_USE_BINARY_COMMUNICATION](#)
=> Resolved in SDHLibrary 0.0.2.2

Implements [SDH::cSerialBase](#).

10.18.3.8 ssize_t SDH::cRS232::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cRS232Exception*) [virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implements [SDH::cSerialBase](#).

10.18.3.9 ssize_t cRS232::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cRS232Exception*) [virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implements [SDH::cSerialBase](#).

10.18.3.10 `char * cRS232::readline (char * line, int size, char * eol, bool return_on_less_data) throw (cRS232Exception*)`

10.18.3.11 `void cRS232::SetTimeout (double timeout) throw (cSerialBaseException*)`
[virtual]

set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

Reimplemented from [SDH::cSerialBase](#).

10.18.3.12 `virtual bool SDH::cRS232::UseCRC16 ()` [inline, virtual]

overloaded from [cSerialBase::UseCRC16](#) since we want to use a CRC16 to protect binary RS232 communication

Reimplemented from [SDH::cSerialBase](#).

10.18.3.13 `virtual bool SDH::cRS232::UseCRC16 ()` [inline, virtual]

overloaded from [cSerialBase::UseCRC16](#) since we want to use a CRC16 to protect binary RS232 communication

Reimplemented from [SDH::cSerialBase](#).

10.18.3.14 `int cRS232::write (char const * ptr, int len = 0) throw (cRS232Exception*)`
[virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the rs232 device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

!! dwWritten is always 0! Damn bloody windows

Implements [SDH::cSerialBase](#).

10.18.3.15 `int SDH::cRS232::write (char const * ptr, int len = 0) throw (cRS232Exception*)`
[virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the rs232 device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

Implements [SDH::cSerialBase](#).

10.18.4 Member Data Documentation

10.18.4.1 unsigned long SDH::cRS232::baudrate [protected]

the baudrate in bit/s

10.18.4.2 std::string SDH::cRS232::device_format_string [protected]

the sprintf format string to generate the device name from the port, see Constructor

10.18.4.3 int SDH::cRS232::fd [protected]

the file descriptor of the RS232 port

10.18.4.4 termios SDH::cRS232::io_set_old [protected]

10.18.4.5 int SDH::cRS232::port [protected]

the RS232 portnumber to use

the RS232 port number to use (port 0 is COM1)

10.18.4.6 int SDH::cRS232::status [protected]

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

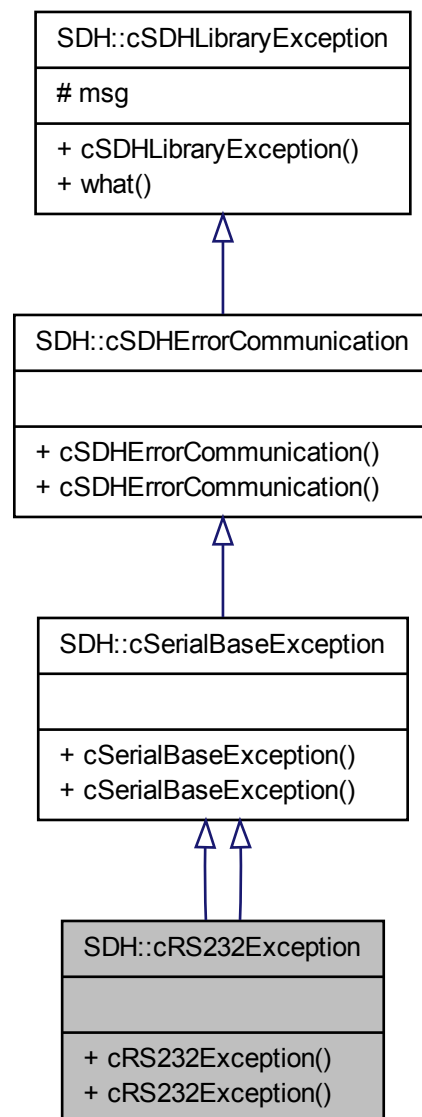
- [sdh/rs232-cygwin.h](#)
- [sdh/rs232-vcc.h](#)
- [sdh/rs232-cygwin.cpp](#)
- [sdh/rs232-vcc.cpp](#)

10.19 SDH::cRS232Exception Class Reference

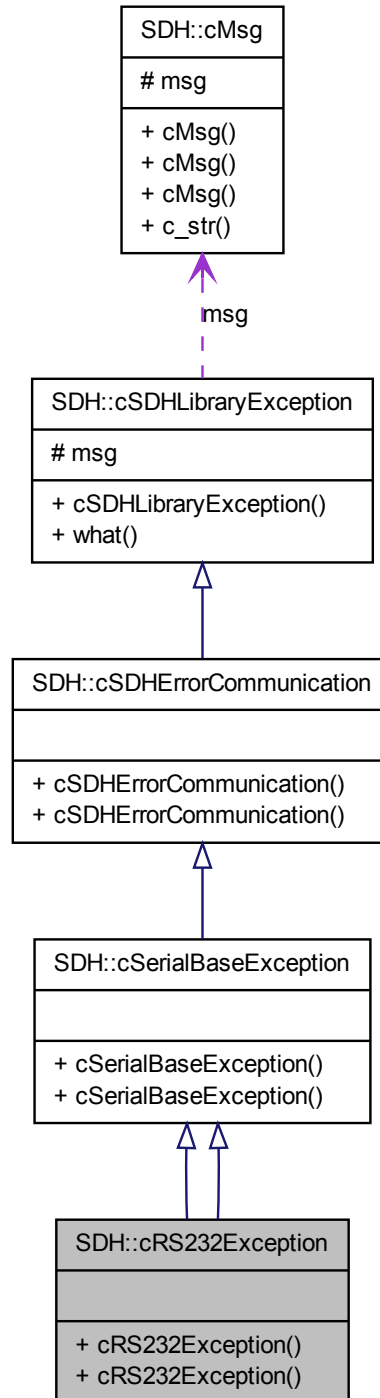
Derived exception class for low-level RS232 related exceptions.

```
#include <rs232-vcc.h>
```

Inheritance diagram for SDH::cRS232Exception:



Collaboration diagram for SDH::cRS232Exception:



Public Member Functions

- [cRS232Exception](#) ([cMsg](#) const &_msg)
- [cRS232Exception](#) ([cMsg](#) const &_msg)

10.19.1 Detailed Description

Derived exception class for low-level RS232 related exceptions.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 SDH::cRS232Exception::cRS232Exception ([cMsg](#) const & *msg*) [[inline](#)]

10.19.2.2 SDH::cRS232Exception::cRS232Exception ([cMsg](#) const & *msg*) [[inline](#)]

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

- [sdh/rs232-cygwin.h](#)
- [sdh/rs232-vcc.h](#)

10.20 SDH::cSDH Class Reference

[SDH::cSDH](#) is the end user interface class to control a SDH (SCHUNK Dexterous Hand).

```
#include <sdh.h>
```

Inheritance diagram for SDH::cSDH:



Collaboration diagram for SDH::cSDH:



Public Types

- enum [eMotorCurrentMode](#) { [eMCM_MOVE](#) = 0, [eMCM_GRIP](#) = 1, [eMCM_HOLD](#) = 2, [eMCM_DIMENSION](#) }
- enum [eAxisState](#) {
[eAS_IDLE](#) = 0, [eAS_POSITIONING](#), [eAS_SPEED_MODE](#), [eAS_NOT_INITIALIZED](#),
[eAS_CW_BLOCKED](#), [eAS_CCW_BLOCKED](#), [eAS_DISABLED](#), [eAS_LIMITS_REACHED](#),
[eAS_DIMENSION](#) }

the motor current can be set specifically for these modes:

The state of an axis (see [TPOSCON_STATE](#) in [global.h](#) of the SDH firmware)

Public Member Functions

- [cSDH](#)(bool _use_radians=false, bool _use_fahrenheit=false, int _debug_level=0)
- virtual [~cSDH](#)()

Constructor of [cSDH](#) class.

Communication methods

- void [OpenRS232](#)(int _port=0, unsigned long _baudrate=115200, double _timeout=-1, char const *_device_format_string="/dev/ttyS%d") throw (cSDHLibraryException*)
- void [OpenCAN_ESD](#)(int _net=0, unsigned long _baudrate=1000000, double _timeout=0.0, [Int32](#) _id_read=43, [Int32](#) _id_write=42) throw (cSDHLibraryException*)
- void [OpenCAN_ESD](#)([tDeviceHandle](#) _ntcan_handle, double _timeout=0.0, [Int32](#) _id_read=43, [Int32](#) _id_write=42) throw (cSDHLibraryException*)
- void [OpenCAN_PEAK](#)(unsigned long _baudrate=1000000, double _timeout=0.0, [Int32](#) _id_read=43, [Int32](#) _id_write=42, const char *_device="/dev/pcanusb0") throw (cSDHLibraryException*)
- void [OpenCAN_PEAK](#)([tDeviceHandle](#) _handle, double _timeout=0.0, [Int32](#) _id_read=43, [Int32](#) _id_write=42) throw (cSDHLibraryException*)
- void [OpenTCP](#)(char const *_tcp_adr="192.168.1.1", int _tcp_port=23, double _timeout=0.0) throw (cSDHLibraryException*)
- void [Close](#)(bool leave_enabled=false) throw (cSDHLibraryException*)
- virtual bool [IsOpen](#)(void) throw ()

Auxiliary movement methods

- void [EmergencyStop](#)(void) throw (cSDHLibraryException*)
- void [Stop](#)(void) throw (cSDHLibraryException*)
- void [SetController](#)([cSDHBase::eControllerType](#) controller) throw (cSDHLibraryException*)
- [eControllerType](#) [GetController](#)(void) throw (cSDHLibraryException*)
- void [SetVelocityProfile](#)([eVelocityProfile](#) velocity_profile) throw (cSDHLibraryException*)

- [eVelocityProfile GetVelocityProfile](#) (void) throw (cSDHLibraryException*)

Methods to access SDH on axis-level

- void [SetAxisMotorCurrent](#) (std::vector< int > const &axes, std::vector< double > const &motor_currents, [eMotorCurrentMode](#) mode=eMCM_MOVE) throw (cSDHLibraryException*)
- void [SetAxisMotorCurrent](#) (int iAxis, double motor_current, [eMotorCurrentMode](#) mode=eMCM_MOVE) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisMotorCurrent](#) (std::vector< int > const &axes, [eMotorCurrentMode](#) mode=eMCM_MOVE) throw (cSDHLibraryException*)
- double [GetAxisMotorCurrent](#) (int iAxis, [eMotorCurrentMode](#) mode=eMCM_MOVE) throw (cSDHLibraryException*)
- void [SetAxisEnable](#) (std::vector< int > const &axes, std::vector< double > const &states) throw (cSDHLibraryException*)
- void [SetAxisEnable](#) (int iAxis=All, double state=1.0) throw (cSDHLibraryException*)
- void [SetAxisEnable](#) (std::vector< int > const &axes, std::vector< bool > const &states) throw (cSDHLibraryException*)
- void [SetAxisEnable](#) (int iAxis=All, bool state=true) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisEnable](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisEnable](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< [eAxisState](#) > [GetAxisActualState](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- [eAxisState GetAxisActualState](#) (int iAxis) throw (cSDHLibraryException*)
- void [WaitAxis](#) (std::vector< int > const &axes, double timeout=-1.0) throw (cSDHLibraryException*)
- void [WaitAxis](#) (int iAxis, double timeout=-1.0) throw (cSDHLibraryException*)
- void [SetAxisTargetAngle](#) (std::vector< int > const &axes, std::vector< double > const &angles) throw (cSDHLibraryException*)
- void [SetAxisTargetAngle](#) (int iAxis, double angle) throw (cSDHLibraryException*)
- std::vector< double > [SetAxisTargetGetAxisActualAngle](#) (std::vector< int > const &axes, std::vector< double > const &angles) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisTargetAngle](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisTargetAngle](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisActualAngle](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisActualAngle](#) (int iAxis) throw (cSDHLibraryException*)
- void [SetAxisTargetVelocity](#) (std::vector< int > const &axes, std::vector< double > const &velocities) throw (cSDHLibraryException*)
- void [SetAxisTargetVelocity](#) (int iAxis, double velocity) throw (cSDHLibraryException*)
- std::vector< double > [SetAxisTargetGetAxisActualVelocity](#) (std::vector< int > const &axes, std::vector< double > const &velocities) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisTargetVelocity](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisTargetVelocity](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisLimitVelocity](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisLimitVelocity](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisLimitAcceleration](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)

- double [GetAxisLimitAcceleration](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisActualVelocity](#) (std::vector< int >const &axes) throw (cSDHLibraryException*)
- double [GetAxisActualVelocity](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisReferenceVelocity](#) (std::vector< int >const &axes) throw (cSDHLibraryException*)
- double [GetAxisReferenceVelocity](#) (int iAxis) throw (cSDHLibraryException*)
- void [SetAxisTargetAcceleration](#) (std::vector< int >const &axes, std::vector< double >const &accelerations) throw (cSDHLibraryException*)
- void [SetAxisTargetAcceleration](#) (int iAxis, double acceleration) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisTargetAcceleration](#) (std::vector< int >const &axes) throw (cSDHLibraryException*)
- double [GetAxisTargetAcceleration](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisMinAngle](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisMinAngle](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisMaxAngle](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisMaxAngle](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisMaxVelocity](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisMaxVelocity](#) (int iAxis) throw (cSDHLibraryException*)
- std::vector< double > [GetAxisMaxAcceleration](#) (std::vector< int > const &axes) throw (cSDHLibraryException*)
- double [GetAxisMaxAcceleration](#) (int iAxis) throw (cSDHLibraryException*)
- double [MoveAxis](#) (std::vector< int >const &axes, bool sequ=true) throw (cSDHLibraryException*)
- double [MoveAxis](#) (int iAxis, bool sequ=true) throw (cSDHLibraryException*)

Methods to access SDH on finger-level

- void [SetFingerEnable](#) (std::vector< int > const &fingers, std::vector< double > const &states) throw (cSDHLibraryException*)
- void [SetFingerEnable](#) (int iFinger, double state=1.0) throw (cSDHLibraryException*)
- void [SetFingerEnable](#) (std::vector< int > const &fingers, std::vector< bool > const &states) throw (cSDHLibraryException*)
- void [SetFingerEnable](#) (int iFinger, bool state) throw (cSDHLibraryException*)
- std::vector< double > [GetFingerEnable](#) (std::vector< int > const &fingers) throw (cSDHLibraryException*)
- double [GetFingerEnable](#) (int iFinger) throw (cSDHLibraryException*)
- void [SetFingerTargetAngle](#) (int iFinger, std::vector< double > const &angles) throw (cSDHLibraryException*)
- void [SetFingerTargetAngle](#) (int iFinger, double a0, double a1, double a2) throw (cSDHLibraryException*)
- std::vector< double > [GetFingerTargetAngle](#) (int iFinger) throw (cSDHLibraryException*)
- void [GetFingerTargetAngle](#) (int iFinger, double &a0, double &a1, double &a2) throw (cSDHLibraryException*)
- std::vector< double > [GetFingerActualAngle](#) (int iFinger) throw (cSDHLibraryException*)
- void [GetFingerActualAngle](#) (int iFinger, double &a0, double &a1, double &a2) throw (cSDHLibraryException*)
- std::vector< double > [GetFingerMinAngle](#) (int iFinger) throw (cSDHLibraryException*)
- void [GetFingerMinAngle](#) (int iFinger, double &a0, double &a1, double &a2) throw (cSDHLibraryException*)

- `std::vector< double > GetFingerMaxAngle (int iFinger) throw (cSDHLibraryException*)`
- `void GetFingerMaxAngle (int iFinger, double &a0, double &a1, double &a2) throw (cSDHLibraryException*)`
- `std::vector< double > GetFingerXYZ (int iFinger, std::vector< double > const &angles) throw (cSDHLibraryException*)`
- `std::vector< double > GetFingerXYZ (int iFinger, double a0, double a1, double a2) throw (cSDHLibraryException*)`
- `double MoveFinger (std::vector< int > const &fingers, bool sequ=true) throw (cSDHLibraryException*)`
- `double MoveFinger (int iFinger, bool sequ=true) throw (cSDHLibraryException*)`
- `double MoveHand (bool sequ=true) throw (cSDHLibraryException*)`

Methods to access %SDH grip skills

- `double GetGripMaxVelocity (void)`
- `double GripHand (eGraspId grip, double close, double velocity, bool sequ=true) throw (cSDHLibraryException*)`

Public Attributes

Predefined index vector objects

- `std::vector< int > all_axes`
A vector with indices of all axes (in natural order), including the virtual axis.
- `std::vector< int > all_real_axes`
A vector with indices of all real axes (in natural order), excluding the virtual axis.
- `std::vector< int > all_fingers`
A vector with indices of all fingers (in natural order)
- `std::vector< int > all_temperature_sensors`
A vector with indices of all temperature sensors.

Predefined unit conversion objects

Pointers to the unit converter objects used by this *cSDH* object.

The referred objects convert values between different unit systems. Example: convert angle values between degrees and radians, temperatures between degrees celsius and degrees fahrenheit or the like.

A *cSDH* object uses these converter objects to convert between external (user) and internal (SDH) units. The user can easily change the converter object that is used for a certain kind of unit. This way a *cSDH* object can easily report and accept parameters in the user or application specific unit system.

Additionally, users can easily add conversion objects for their own, even more user- or application-specific unit systems.

- `const cUnitConverter * uc_angle`
*unit convert for (axis) angles: default = *SDH::cSDH::uc_angle_degrees**

- `const cUnitConverter * uc_angular_velocity`
unit convert for (axis) angular velocities: default = `SDH::cSDH::uc_angular_velocity_degrees_per_second`
- `const cUnitConverter * uc_angular_acceleration`
unit convert for (axis) angular accelerations: default = `SDH::cSDH::uc_angular_acceleration_degrees_per_second_squared`
- `const cUnitConverter * uc_time`
unit convert for times: default = `uc_time_seconds`
- `const cUnitConverter * uc_temperature`
unit convert for temperatures: default = `SDH::cSDH::uc_temperature_celsius`
- `const cUnitConverter * uc_motor_current`
unit converter for motor current: default = `SDH::cSDH::uc_motor_current_ampere`
- `const cUnitConverter * uc_position`
unit converter for position: default = `SDH::cSDH::uc_position_millimeter`

Static Public Attributes

Predefined unit conversion objects

Some predefined `cUnitConverter` unit conversion objects to convert values between different unit systems. These are static members since the converter objects do not depend on the individual `cSDH` object.

For every physical unit used in the `cSDH` class there is at least one (most of the time more than one) predefined unit converter. For example for angles there are radians and degrees.

- `static cUnitConverter const uc_angle_degrees`
Default converter for angles (internal unit == external unit): degrees.
- `static cUnitConverter const uc_angle_radians`
Converter for angles: external unit = radians.
- `static cUnitConverter const uc_time_seconds`
Default converter for times (internal unit == external unit): seconds.
- `static cUnitConverter const uc_time_milliseconds`
Converter for times: external unit = milliseconds.
- `static cUnitConverter const uc_temperature_celsius`
Default converter for temperatures (internal unit == external unit): degrees celsius.
- `static cUnitConverter const uc_temperature_fahrenheit`
Converter for temperatures: external unit = degrees fahrenheit.

- static [cUnitConverter](#) const [uc_angular_velocity_degrees_per_second](#)
Default converter for angular velocities (internal unit == external unit): degrees / second.
- static [cUnitConverter](#) const [uc_angular_velocity_radians_per_second](#)
Converter for angular velocities: external unit = radians/second.
- static [cUnitConverter](#) const [uc_angular_acceleration_degrees_per_second_squared](#)
Default converter for angular accelerations (internal unit == external unit): degrees / second.
- static [cUnitConverter](#) const [uc_angular_acceleration_radians_per_second_squared](#)
Converter for angular velocities: external unit = radians/second.
- static [cUnitConverter](#) const [uc_motor_current_ampere](#)
Default converter for motor current (internal unit == external unit): Ampere.
- static [cUnitConverter](#) const [uc_motor_current_milliampere](#)
Converter for motor current: external unit = milli Ampere.
- static [cUnitConverter](#) const [uc_position_millimeter](#)
Default converter for position (internal unit == external unit): millimeter.
- static [cUnitConverter](#) const [uc_position_meter](#)
Converter for position: external unit = meter.

Protected Member Functions

Internal helper methods

- `std::vector< double > SetAxisValueVector (std::vector< int > const &axes, std::vector< double > const &values, pSetFunction ll_set, pGetFunction ll_get, cUnitConverter const *uc, std::vector< double > const &min_values, std::vector< double > const &max_values, char const *name) throw (cSDHLibraryException*)`
- `std::vector< double > GetAxisValueVector (std::vector< int > const &axes, pGetFunction ll_get, cUnitConverter const *uc, char const *name) throw (cSDHLibraryException*)`
- `std::vector< int > ToIndexVector (int index, std::vector< int > &all_replacement, int maxindex, char const *name) throw (cSDHLibraryException*)`
- `pSetFunction GetMotorCurrentModeFunction (eMotorCurrentMode mode) throw (cSDHLibraryException*)`
- `std::vector< double > _GetFingerXYZ (int fi, std::vector< double > r_angles) throw (cSDHLibraryException*)`

Protected Attributes

- int [NUMBER_OF_AXES_PER_FINGER](#)

The number of axis per finger (for finger 1 this includes the "virtual" base axis)

- `int` [NUMBER_OF_VIRTUAL_AXES](#)
The number of virtual axes.
- `int` [nb_all_axes](#)
The number of all axes including virtual axes.
- `std::vector< int >` [finger_number_of_axes](#)
Mapping of finger index to number of real axes of fingers:
- `std::vector< std::vector< int > >` [finger_axis_index](#)
Mapping of finger index, finger axis index to axis index:
- `std::vector< double >` [f_zeros_v](#)
Vector of 3 epsilon values.
- `std::vector< double >` [f_ones_v](#)
Vector of 3 1.0 values.
- `std::vector< double >` [zeros_v](#)
Vector of nb_all_axes 0.0 values.
- `std::vector< double >` [ones_v](#)
Vector of nb_all_axes 1.0 values.
- `std::vector< double >` [f_min_motor_current_v](#)
Minimum allowed motor currents (in internal units (Ampere)), including the virtual axis.
- `std::vector< double >` [f_max_motor_current_v](#)
Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.
- `std::vector< double >` [f_min_angle_v](#)
Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.
- `std::vector< double >` [f_max_angle_v](#)
Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.
- `std::vector< double >` [f_min_velocity_v](#)
Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.
- `std::vector< double >` [f_max_velocity_v](#)
Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

- `std::vector< double > f_min_acceleration_v`
*Minimum allowed axis acceleration (in internal units (degrees/(second * second))), including the virtual axis.*
- `std::vector< double > f_max_acceleration_v`
*Maximum allowed axis acceleration (in internal units (degrees/(second * second))), including the virtual axis.*
- `double grip_max_velocity`
Maximum allowed grip velocity (in internal units (degrees/second))

Kinematic parameters of the Hand

- `double l1`
length of limb 1 (proximal joint to distal joint) in mm
- `double l2`
length of limb 2 (distal joint to fingertip) in mm
- `double d`
- `double h`
- `std::vector< std::vector< double > > offset`
- `cSerialBase * com`
- `cSDHSerial comm_interface`
The object to interface with the SDH attached via serial RS232 or CAN or TCP.
- `virtual void SetDebugOutput (std::ostream *debuglog)`
change the stream to use for debug messages

Miscellaneous methods

- `bool IsVirtualAxis (int iAxis) throw (cSDHLibraryException*)`
Return true if index iAxis refers to a virtual axis.
- `void UseRadians (void)`
- `void UseDegrees (void)`
- `int GetFingerNumberOfAxes (int iFinger) throw (cSDHLibraryException*)`
- `int GetFingerAxisIndex (int iFinger, int iFingerAxis) throw (cSDHLibraryException*)`
- `char const * GetFirmwareRelease (void) throw (cSDHLibraryException*)`
- `bool CheckFirmwareRelease (void) throw (cSDHLibraryException*)`
- `char const * GetInfo (char const *what) throw (cSDHLibraryException*)`
- `std::vector< double > GetTemperature (std::vector< int > const &sensors) throw (cSDHLibraryException*)`

- double [GetTemperature](#) (int iSensor) throw (cSDHLibraryException*)
- static char const * [GetLibraryRelease](#) (void)
- static char const * [GetLibraryName](#) (void)
- static char const * [GetFirmwareReleaseRecommended](#) (void)

10.20.1 Detailed Description

[SDH::cSDH](#) is the end user interface class to control a SDH (SCHUNK Dexterous Hand). A general overview of the structure and architecture used is given [here](#).

Remarks

- The [cSDH](#) class provides methods to access the 7 axes of the SDH individually as well as on a finger level.
 - When accessing the axes individually then the following axis indices must be used to address an axis / some axes:
 - * 0 : common base axis of finger 0 and 2
 - * 1 : proximal axis of finger 0
 - * 2 : distal axis of finger 0
 - * 3 : proximal axis of finger 1
 - * 4 : distal axis of finger 1
 - * 5 : proximal axis of finger 2
 - * 6 : distal axis of finger 2
 - When accessing the axes on finger level then every finger has 3 axes for a uniform interface of the access methods. Her the following finger axis indices must be used:
 - * 0 : base axis of finger (for finger 1 this is a "virtual" axis with min angle = max angle = 0.0)
 - * 1 : proximal axis of finger
 - * 2 : distal axis of finger
- Vector-like parameters: The interface functions defined here make full use of the flexibility provided by the STL `vector<T>` type. I.E. for parameters of functions like axis indices or axis angles not only single numerical values can be given, but also vectors of `int` or `double` values. This way the same (overloaded) interface function can address a single axis individually or multiple axes in a call, as required by the application. Such parameters are herein referred to as "vectors".
- Parameters for methods are checked for validity. In case an invalid parameter is given the method throws a [cSDHErrorInvalidParameter](#) exception.
- The underlying physical unit system of parameters that do have a unit (like angles, velocities or temperatures) can be adapted to the users or the applications need. See also [unit conversion](#) objects". The default converter objects are set as the `uc_*` member variables ([uc_angle](#), [uc_angular_velocity](#), [uc_angular_acceleration](#), [uc_time](#), [uc_temperature](#), [uc_position](#)). The units are changed in the communication between user application and [cSDH](#) object instance only (USERAPP and SDHLibrary-CPP in the [overview](#) figure"). For now the SDH firmware knows only about its internal unit system.

10.20.2 Member Enumeration Documentation

10.20.2.1 enum SDH::cSDH::eAxisState

The state of an axis (see TPOSCON_STATE in global.h of the SDH firmware)

Enumerator:

- eAS_IDLE* axis is idle
- eAS_POSITIONING* the goal position has not been reached yet
- eAS_SPEED_MODE* axis is in speed mode
- eAS_NOT_INITIALIZED* axis is not initialized or doesn't exist
- eAS_CW_BLOCKED* axis is blocked in counterwise direction
- eAS_CCW_BLOCKED* axis is blocked is blocked in against counterwise direction
- eAS_DISABLED* axis is disabled
- eAS_LIMITS_REACHED* position limits reached and axis stopped
- eAS_DIMENSION* Endmarker and Dimension.

10.20.2.2 enum SDH::cSDH::eMotorCurrentMode

the motor current can be set specifically for these modes:

Enumerator:

- eMCM_MOVE* The motor currents used while "moving" with a [MoveHand\(\)](#) or [MoveFinger\(\)](#) command.
- eMCM_GRIP* The motor currents used while "gripping" with a [GripHand\(\)](#) command.
- eMCM_HOLD* The motor currents used after "gripping" with a [GripHand\(\)](#) command (i.e. "holding")
- eMCM_DIMENSION* Endmarker and Dimension.

10.20.3 Constructor & Destructor Documentation

10.20.3.1 cSDH::cSDH (bool *_use_radians* = false, bool *_use_fahrenheit* = false, int *_debug_level* = 0)

Constructor of [cSDH](#) class.

Creates an new object of type [cSDH](#). One such object is needed for each SDH that you want to control. The constructor initializes internal data structures. A connection the SDH is **not** yet established, see [OpenRS232\(\)](#) on how to do that.

After an object is created the user can adjust the unit systems used to set/report parameters to/from SDH. This is shown in the example code below. The default units used (if not overwritten by constructor parameters) are:

- degrees [deg] for (axis) angles
- degrees per second [deg/s] for (axis) angular velocities
- seconds [s] for times
- degrees celsius [deg C] for temperatures

Parameters

<code>_use_radians</code>	: Flag, if true then use radians and radians/second to set/report (axis) angles and angular velocities instead of default degrees and degrees/s.
<code>_use_fahrenheit</code>	: Flag, if true then use degrees fahrenheit to report temperatures instead of default degrees celsius.
<code>_debug_level</code>	: The level of debug messages to print <ul style="list-style-type: none"> • 0: (default) no messages • 1: messages of this cSDH instance • 2: like 1 plus messages of the inner cSDHSerial instance

Examples:

Common use:

```
// Include the cSDH interface
#include <sdh.h>

// Create a cSDH object 'hand'.
cSDH hand();
```

The mentioned change of a unit system can be done like this:

```
// Assuming 'hand' is a cSDH object ...

// override default unit converter for (axis) angles:
hand.uc_angle = &cSDH::uc_angle_radians;

// override default unit converter for (axis) angular velocities:
hand.uc_angular_velocity = &
cSDH::uc_angular_velocity_radians_per_second;

// override default unit converter for (axis) angular accelerations:
hand.uc_angular_acceleration = &
cSDH::uc_angular_acceleration_radians_per_second_squared;

// instead of the last 3 calls the following shortcut could be used:
hand.UseRadians();

// override default unit converter for times:
hand.uc_time = &cSDH::uc_time_milliseconds;

// override default unit converter for temperatures:
hand.uc_temperature = &cSDH::uc_temperature_fahrenheit;

// override default unit converter for positions:
hand.uc_position = &cSDH::uc_position_meter;
```

For convenience the most common settings can be specified as bool parameters for the constructor, like in:

```
// Include the cSDH interface
#include <sdh.h>

// Create a cSDH object 'hand' that uses
// - the non default radians and radians/s units,
// - the default temperature in degrees celsius,
// - A debug level of 2
cSDH hand( true, false, 2 );
```

unit convert for times: default = `uc_time_seconds`

unit convert for temperatures: default = `uc_temperature_celsius`

unit converter for motor current: default = `uc_motor_current_ampere`

unit converter for position: default = `uc_position_millimeter`

The number of axis per finger (for finger 1 this includes the "virtual" base axis)

The number of virtual axes

Mapping of finger index to number of real axes of fingers:

Mapping of finger index, finger axis index to axis index:

Maximum allowed grip velocity (in internal units (degrees/second))

10.20.3.2 cSDH::~~cSDH() [virtual]

Virtual destructor to make compiler happy

If the connection to the SDH hardware/firmware is still open then the connection is closed, which will stop the axis controllers (and thus prevent overheating).

10.20.4 Member Function Documentation

10.20.4.1 std::vector< double > cSDH::GetFingerXYZ(int fi, std::vector< double > r_angles) throw (cSDHLibraryException*) [protected]

return cartesian [x,y,z] position in mm of fingertip for finger fi at angles r_angles (rad)

10.20.4.2 bool cSDH::CheckFirmwareRelease(void) throw (cSDHLibraryException*)

Check the actual release of the firmware of the connected SDH against the recommended firmware release.

Returns

true - if the actual firmware is the recommended one false - the actual firmware is NOT the recommended one (communication with the [SDH](#) might not work as expected)

This will throw a (`cSDHErrorCommunication*`) exception if the connection to the SDH is not yet opened.

Examples:

```
// Assuming 'hand' is a cSDH object ...

if ( hand.CheckFirmwareRelease() )
{
    cout << "The firmware release of the connected SDH is the one recommended by this SDHLibrary\n";
}
else
{
    cout << "The firmware release of the connected SDH is NOT the one recommended by this SDHLibrary\n";
    cout << "  Actual SDH firmware release      " << hand.GetFirmwareRelease() << "\n";
    cout << "  Recommended SDH firmware release " << hand.GetFirmwareReleaseRecommended() << "\n";
}
```

See also

See [GetFirmwareReleaseRecommended\(\)](#) to get the recommended SDH firmware release.

10.20.4.3 void cSDH::Close (bool *leave_enabled* = false) throw (`cSDHLibraryException*`)

Close connection to SDH.

The default behaviour is to **not** leave the controllers of the SDH enabled (to prevent overheating). To keep the controllers enabled (e.g. to keep the finger axes actively in position) set *leave_enabled* to `true`. Only already enabled axes will be left enabled.

Parameters

<i>leave_enabled</i>	- Flag: true to leave the controllers on, false (default) to disable the controllers (switch powerless)
----------------------	---

This throws a (`cSDHErrorCommunication*`) exception if the connection was not opened before.

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Close connection to SDH, power off controllers:
hand.Close();

// To leave the already enabled controllers enabled:
hand.Close( true );
```

10.20.4.4 void cSDH::EmergencyStop (void) throw (cSDHLibraryException*)

Stop movement of all axes of the SDH and switch off the controllers

This command will always be executed sequentially: it will return only after the SDH has confirmed the emergency stop.

Bug

For now this will **NOT** work while a [GripHand\(\)](#) command is executing, even if that was initiated non-sequentially!

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Perform an emergency stop:
hand.EmergencyStop();
```

10.20.4.5 std::vector< double > cSDH::GetAxisActualAngle (std::vector< int > const & axes) throw (cSDHLibraryException*)

Get the current actual angle(s) of axis(axes).

The actual angles are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the actual angles of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angle](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisActualAngle\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get actual axis angle of all axes
std::vector<double> v = hand.GetAxisActualAngle( hand.all_axes );
```

```
// now v is something like {0.0, 0.0, 42.0, 0.0, 47.11, 0,0, 0.0}

// Get actual axis angle of axis 2
double v2 = hand.GetAxisActualAngle( 2 );
// 2 is now something like 42.0

// Get actual axis angle of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisActualAngle( axes24 );
// now v is something like {42.0, 47.11}
```

10.20.4.6 double cSDH::GetAxisActualAngle (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisActualAngle\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single angle, see there for details and examples.

10.20.4.7 std::vector< cSDH::eAxisState > cSDH::GetAxisActualState (std::vector< int > const & *axes*) throw (cSDHLibraryException*)

Get the current actual state(s) of axis(*axes*).

The actual axis states are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the actual states of the selected axes.
- The values are given as [eAxisState](#) enum values
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisActualState\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get actual axis state of all axes
std::vector<eAxisState> v = hand.GetAxisActualState( hand.all_axes )
// now v is something like {eAS_IDLE, eAS_POSITIONING, eAS_IDLE, eAS_ID
```

```

LE, eAS_IDLE, eAS_DISABLED, eAS_IDLE}

// Get actual axis state of axis 3
eAxisState v3 = hand.GetAxisActualState( 3 );
// v3 is now something like eAS_IDLE

// Get actual state of axis 2 and 5
std::vector<int> axes25;
axes25.push_back( 2 );
axes25.push_back( 5 );

v = hand.GetAxisActualState( axes25 );
// now v is something like {eAS_IDLE, eAS_DISABLED}

```

10.20.4.8 cSDH::eAxisState cSDH::GetAxisActualState (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisActualState\(std::vector<int>const&\)](#), just for a single axis *iAxis*, see there for details and examples.

10.20.4.9 std::vector< double > cSDH::GetAxisActualVelocity (std::vector< int >const & axes) throw (cSDHLibraryException*)

Get the actual velocity(s) of axis(axes).

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the actual velocities of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_velocity](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisActualVelocity\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```

// Assuming "hand" is a cSDH object ...

// Get actual axis velocity of all axes
std::vector<double> v = hand.GetAxisActualVelocity( hand.all_axes );

```

```
// now v is something like {0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7}

// Get actual axis velocity of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );
v = hand.GetAxisActualVelocity( axes24 );
// now v is something like {13.2, 0.0}

// Get actual axis velocity of axis 2
double v3 = hand.GetAxisActualVelocity( 2 );
// v3 is now something like 13.2
```

10.20.4.10 `double cSDH::GetAxisActualVelocity (int iAxis) throw (cSDHLibraryException*)`

Like [GetAxisActualVelocity\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single velocity, see there for details and examples.

10.20.4.11 `double cSDH::GetAxisEnable (int iAxis) throw (cSDHLibraryException*)`

Like [GetAxisEnable\(std::vector<int>const&\)](#), just for a single axis *iAxis*, see there for details and examples.

10.20.4.12 `std::vector< double > cSDH::GetAxisEnable (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get enabled/disabled state of axis controller(s).

The enabled/disabled state of the controllers of the selected axes is read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of enabled/disabled states as doubles (0=disabled, 1.0=enabled) of the selected axes.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisEnable\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming 'hand' is a cSDH object ...
```



```

// Get enabled state of all axes:
std::vector<double> v = hand.GetAxisEnable( hand.all_axes );
// now v is something like {0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0}

// Get enabled state of axis 3 and 5
std::vector<int> axes35;
axes35.push_back( 3 );
axes35.push_back( 5 );

v = hand.GetAxisEnable( axes35 );
// now v is something like {1.0, 0.0}

// Get enabled state of axis 3
double v3 = hand.GetAxisEnable( 3 );
// now v3 is something like 1.0

```

10.20.4.13 `std::vector< double > cSDH::GetAxisLimitAcceleration (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the acceleration limit(s) of axis(axes).

The acceleration limit(s) are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the acceleration limits of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_acceleration](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisLimitAcceleration\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```

// Assuming "hand" is a cSDH object ...

// Get axis acceleration limits of all axes
std::vector<double> v = hand.GetAxisLimitAcceleration( hand.all_axes );

// now v is something like {81.0, 140.0, 120.0, 140.0, 120.0, 140.0, 12
0.0}

```

```
// Get axis acceleration limit of axis 2
double v2 = hand.GetAxisLimitAcceleration( 2 );
// v2 is now something like 120.0

// Get axis acceleration limits of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisLimitAcceleration( axes24 );
// now v is something like {120.0,120.0}
```

10.20.4.14 `double cSDH::GetAxisLimitAcceleration (int iAxis) throw (cSDHLibraryException*)`

Like [GetAxisLimitAcceleration\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single acceleration limit, see there for details and examples.

10.20.4.15 `double cSDH::GetAxisLimitVelocity (int iAxis) throw (cSDHLibraryException*)`

Like [GetAxisLimitVelocity\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single velocity limit, see there for details and examples.

10.20.4.16 `std::vector< double > cSDH::GetAxisLimitVelocity (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the velocity limit(s) of axis(axes).

The velocity limit(s) are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the velocity limits of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_velocity](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisLimitVelocity\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get axis velocity limits of all axes
std::vector<double> v = hand.GetAxisLimitVelocity( hand.all_axes );
// now v is something like {81.0, 140.0, 120.0, 140.0, 120.0, 140.0, 12
0.0}

// Get axis velocity limit of axis 2
double v2 = hand.GetAxisLimitVelocity( 2 );
// v2 is now something like 120.0

// Get axis velocity limits of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisLimitVelocity( axes24 );
// now v is something like {120.0,120.0}
```

10.20.4.17 `std::vector< double > cSDH::GetAxisMaxAcceleration (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the maximum acceleration(s) of axis(axes).

The maximum accelerations are currently not read from the SDH, but are stored in the library.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the max angular accelerations of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_acceleration](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisMaxAcceleration\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum axis angular accelerations of all axes
```

```

std::vector<double> v = hand.GetAxisMaxAcceleration( hand.all_axes );
// now v is something like {1000.0, 1000.0, 1000.0, 1000.0, 1000.0, 100
0.0, 1000.0}

// Get maximum axis angular acceleration of axis 3
double v3 = hand.GetAxisMaxAcceleration( 3 );
// v3 is now something like 1000.0

// Get maximum axis angular acceleration of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisMaxAcceleration( axes24 );
// now v is something like {1000.0, 1000.0}

// Or if you change the angular acceleration unit system:
hand.UseRadians();

v = hand.GetAxisMaxAcceleration( hand.all_axes );
// now v is something like {17.453292519943293, 17.453292519943293, 17.
453292519943293, 17.453292519943293, 17.453292519943293, 17.453292519943293, 17.4
53292519943293}

```

10.20.4.18 double cSDH::GetAxisMaxAcceleration (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisMaxAcceleration\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single minimum angle, see there for details and examples.

10.20.4.19 double cSDH::GetAxisMaxAngle (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisMaxAngle\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single maximum angle, see there for details and examples.

10.20.4.20 std::vector< double > cSDH::GetAxisMaxAngle (std::vector< int > const & axes) throw (cSDHLibraryException*)

Get the maximum angle(s) of axis(axes).

The maximum angles are currently not read from the SDH, but are stored in the library.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the max angles of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angle](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisMaxAngle\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum axis angles of all axes
std::vector<double> v = hand.GetAxisMaxAngle( hand.all_axes );
// now v is something like {90.0, 90.0, 90.0, 90.0, 90.0, 90.0, 90.0}

// Get maximum axis angle of axis 3
double v3 = hand.GetAxisMaxAngle( 3 );
// v3 is now something like 90.0

// Get maximum axis angle of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisMaxAngle( axes24 );
// now v is something like {90.0, 90.0}

// Or if you change the angle unit system:
hand.UseRadians();

v = hand.GetAxisMaxAngle( hand.all_axes );
// now v is something like { 1.5707963267948966, 1.5707963267948966, 1.
5707963267948966, 1.5707963267948966, 1.5707963267948966, 1.5707963267948966, 1.5
707963267948966 }
```

10.20.4.21 double cSDH::GetAxisMaxVelocity (int iAxis) throw (cSDHLibraryException*)

Like [GetAxisMaxVelocity\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single minimum angle, see there for details and examples.

10.20.4.22 std::vector< double > cSDH::GetAxisMaxVelocity (std::vector< int > const & axes) throw (cSDHLibraryException*)

Get the maximum velocity(s) of axis(axes). These are the (theoretical) maximum velocities as determined by the maximum motor velocity and gear box ratio. The values do not take things like friction or inertia into account. So it is likely that these maximum velocities cannot be reached by the real hardware in reality.

The maximum velocities are currently read once from the SDH when the communication to the SDH is opened. Later queries of this maximum velocities will use the values stored in the library.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the max angular velocities of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_velocity](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisMaxVelocity\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum axis angular velocities of all axes
std::vector<double> v = hand.GetAxisMaxVelocity( hand.all_axes );
// now v is something like {83.857,200.000,157.895,200.000,157.895,200.000,157.895}

// Get maximum axis angular velocity of axis 3
double v3 = hand.GetAxisMaxVelocity( 3 );
// v3 is now something like 200.0

// Get maximum axis angular velocity of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisMaxVelocity( axes24 );
// now v is something like {157.895, 157.895}

// Or if you change the angular velocity unit system:
hand.UseRadians();

v = hand.GetAxisMaxVelocity( hand.all_axes );
// now v is something like {1.46358075084, 3.49065850399, 2.75578762244, 3.49065850399, 2.75578762244, 3.49065850399, 2.75578762244}
```

10.20.4.23 `std::vector< double > cSDH::GetAxisMinAngle (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the minimum angle(s) of axis(axes).

The minimum angles are currently not read from the SDH, but are stored in the library.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the min angles of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angle](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisMinAngle\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get minimum axis angles of all axes
std::vector<double> v = hand.GetAxisMinAngle( hand.all_axes );
// now v is something like {0.0, -90.0, -90.0, -90.0, -90.0, -90.0, -90
.0}

// Get minimum axis angle of axis 3
double v3 = hand.GetAxisMinAngle( 3 );
// v3 is now something like -90.0

// Get minimum axis angle of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisMinAngle( axes24 );
// now v is something like {-90.0, -90.0}

// Or if you change the angle unit system:
hand.UseRadians();

v = hand.GetAxisMinAngle( hand.all_axes );
// now v is something like {0.0, -1.5707963267948966, -1.57079632679489
66, -1.5707963267948966, -1.5707963267948966, -1.5707963267948966, -1.57079632679
48966}
```

10.20.4.24 double cSDH::GetAxisMinAngle (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisMinAngle\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single minimum angle, see there for details and examples.

10.20.4.25 `std::vector< double > cSDH::GetAxisMotorCurrent (std::vector< int > const & axes, eMotorCurrentMode mode = eMCM_MOVE) throw (cSDHLibraryException*)`

Get the maximum allowed motor current(s) of axis(axes).

The maximum allowed motor currents are read from the SDH. The motor currents are stored:

- axis specific
- mode specific (see eMotorCurrentMode)

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>mode</i>	- the mode to set the maximum motor current for. One of the eMotorCurrentMode modes.

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the motor currents of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_motor_current](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisMotorCurrent\(int,eMotorCurrentMode\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum allowed motor currents of all axes
std::vector<double> v = hand.GetAxisMotorCurrent( hand.all_axes );
// now v is something like {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7}

// Get maximum allowed motor current of axis 3 in mode "eMCM_MOVE"
double mc3 = hand.GetAxisMotorCurrent( 3, eMCM_MOVE );
// mc3 is now something like 0.75

// Get maximum allowed motor current of axis 3 and 5 in mode "eMCM_GRIP"
"
std::vector<int> axes35;
axes35.push_back( 3 );
axes35.push_back( 5 );

v = hand.GetAxisMotorCurrent( axes35, eMCM_GRIP );
// now v is something like {0.5,0.5};
```


10.20.4.26 `double cSDH::GetAxisMotorCurrent (int iAxis, eMotorCurrentMode mode = eMCM_MOVE) throw (cSDHLibraryException*)`

Like [GetAxisMotorCurrent\(std::vector<int>const&,eMotorCurrentMode\)](#), just for a single axis, see there for details and examples.

10.20.4.27 `std::vector< double > cSDH::GetAxisReferenceVelocity (std::vector< int >const & axes) throw (cSDHLibraryException*)`

Get the current reference velocity(s) of axis(axes). (This velocity is used internally by the SDH in eCT_VELOCITY_ACCELERATION mode).

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the reference velocities of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_velocity](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisReferenceVelocity\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Switch to "velocity control with acceleration ramp" controller mode
first.
// (When in another controller mode like the default eCT_POSE,
// then the reference velocities will not be valid):
hand.SetController( eCT_VELOCITY_ACCELERATION );

// Get reference axis velocity of all axes
std::vector<double> v = hand.GetAxisReferenceVelocity( hand.all_axes );

// now v is something like {0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7}

// Get reference axis velocity of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );
v = hand.GetAxisReferenceVelocity( axes24 );
// now v is something like {13.2, 0.0}
```

```
// Get reference axis velocity of axis 2
double v3 = hand.GetAxisReferenceVelocity( 2 );
// v3 is now something like 13.2
```

Remarks

- the underlying rvel command of the SDH firmware is not available in firmwares prior to 0.0.2.6. For such hands calling rvel will fail miserably.
- The availability of an appropriate SDH firmware is **not** checked here due to performance losses when this function is used often.

10.20.4.28 `double cSDH::GetAxisReferenceVelocity (int iAxis) throw (cSDHLibraryException*)`

Like [GetAxisReferenceVelocity\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single velocity, see there for details and examples.

10.20.4.29 `std::vector< double > cSDH::GetAxisTargetAcceleration (std::vector< int >const & axes) throw (cSDHLibraryException*)`

Get the target acceleration(s) of axis(axes).

The currently set target accelerations are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the target accelerations of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_acceleration](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisTargetAcceleration\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get target axis acceleration of all axes
std::vector<double> v = hand.GetAxisTargetAcceleration( hand.all_axes )
```

```

;
    // now v is something like {100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 1
00.0}

    // Get target axis acceleration of axis 2
    double v2 = hand.GetAxisTargetAcceleration( 2 );
    // v2 is now something like 100.0

    // Get target axis acceleration of axis 2 and 4
    std::vector<int> axes24;
    axes24.push_back( 2 );
    axes24.push_back( 4 );

    v = hand.GetAxisTargetAcceleration( axes24 );
    // now v is something like {100.0, 100.0}

```

10.20.4.30 double cSDH::GetAxisTargetAcceleration (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisTargetAcceleration\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single acceleration, see there for details and examples.

10.20.4.31 double cSDH::GetAxisTargetAngle (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisTargetAngle\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single angle, see there for details and examples.

10.20.4.32 std::vector< double > cSDH::GetAxisTargetAngle (std::vector< int > const & *axes*) throw (cSDHLibraryException*)

Get the target angle(s) of axis(axes).

The currently set target angles are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the target angles of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angle](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisTargetAngle\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get target axis angle of all axes
std::vector<double> v = hand.GetAxisTargetAngle( hand.all_axes );
// now v is something like {0.0, 0.0, 42.0, 0.0, 47.11, 0.0, 0.0}

// Get target axis angle of axis 2
double v2 = hand.GetAxisTargetAngle( 2 );
// v2 is now something like 42.0

// Get target axis angle of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisTargetAngle( axes24 );
// now v is something like {42.0, 47.11}
```

10.20.4.33 `std::vector< double > cSDH::GetAxisTargetVelocity (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the target velocity(s) of axis(axes).

The currently set target velocities are read from the SDH.

Parameters

<i>axes</i>	- A vector of axis indices to access.
-------------	---------------------------------------

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the target velocities of the selected axes.
- The values are converted to the selected external unit system using the configured [uc_angular_velocity](#) unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisTargetVelocity\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...
```

```

// Get target axis velocity of all axes
std::vector<double> v = hand.GetAxisTargetVelocity( hand.all_axes );
// now v is something like {0.0, 0.0, 42.0, 0.0, 47.11, 0.0, 0.0}

// Get target axis velocity of axis 2
double v2 = hand.GetAxisTargetVelocity( 2 );
// v2 is now something like 42.0

// Get target axis velocity of axis 2 and 4
std::vector<int> axes24;
axes24.push_back( 2 );
axes24.push_back( 4 );

v = hand.GetAxisTargetVelocity( axes24 );
// now v is something like {42.0, 47.11}

```

10.20.4.34 double cSDH::GetAxisTargetVelocity (int *iAxis*) throw (cSDHLibraryException*)

Like [GetAxisTargetVelocity\(std::vector<int>const&\)](#), just for a single axis *iAxis* and returning a single velocity, see there for details and examples.

10.20.4.35 std::vector< double > cSDH::GetAxisValueVector (std::vector< int > const & *axes*, pGetFunction *ll_get*, cUnitConverter const * *uc*, char const * *name*) throw (cSDHLibraryException*) [protected]

Generic get function: get some given axes values

Parameters

<i>axes</i>	- a vector of axis indices
<i>ll_get</i>	- a pointer to the low level get function to use
<i>uc</i>	- a pointer to the unit converter object to apply before returning values
<i>name</i>	- a string with the name of the values (for constructing error message)

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the addressed values for the selected axes.
- The values are converted to external unit system using the *uc* unit converter object.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

10.20.4.36 `cSDHBase::eControllerType cSDH::GetController (void) throw (cSDHLibraryException*)`

Get the type of axis controller used in the SDH

The currently set controller type will be queried and returned (One of `eControllerType`)

Examples:

```
// Assuming 'hand' is a sdh.cSDH object ...

// Get the controller type of the attached SDH:
ct = hand.GetController();

// Print result, numerically and symbolically
std::cout << "Currently the axis controller type is set to " << ct;
std::cout << "(" << GetStringFromControllerType(ct) << ")\n";
```

10.20.4.37 `std::vector< double > cSDH::GetFingerActualAngle (int iFinger) throw (cSDHLibraryException*)`

Get the current actual axis angles of a single finger.

The current actual axis angles of finger *iFinger* are read from the SDH.

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index.
----------------	---

Remarks

- The *iFinger* index is checked if it is a valid finger index.
- If *iFinger* is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A list of the current actual axis angles of the selected finger
- The values are returned in the configured angle unit system [uc_angle](#).

See also [GetFingerActualAngle\(int,double&,double&,double&\)](#) for an overloaded variant to get finger axis actual angles into single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get actual axis angles of finger 0
std::vector<double> v = hand.GetFingerActualAngle( 0 );
// v is now something like {42.0, -10.0, 47.11}

// Get actual axis angles of finger 1
double a0, a1, a2;
hand.GetFingerTargetAngle( 1, a0, a1, a2 );
// now a0, a1, a2 are something like 0.0, 24.7 and -5.5 respectively.
```

10.20.4.38 void cSDH::GetFingerActualAngle (int *iFinger*, double & *a0*, double & *a1*, double & *a2*) throw (cSDHLibraryException*)

Like [GetFingerActualAngle\(int\)](#), just returning the actual axis angles in the *a0*, *a1* and *a2* parameters which are given by reference.

10.20.4.39 int cSDH::GetFingerAxisIndex (int *iFinger*, int *iFingerAxis*) throw (cSDHLibraryException*)

Return axis index of *iFingerAxis* axis of finger with index *iFinger*

For *iFinger*=2, *iFingerAxis*=0 this will return the index of the virtual base axis of the finger

Parameters

<i>iFinger</i>	- index of finger in range [0..NUMBER_OF_FINGERS-1]
<i>iFingerAxis</i>	- index of finger axis in range [0..NUMBER_OF_AXES_PER_FINGER-1]

Returns

- Axis index of *iFingerAxis*-th axis of finger with index *iFinger*
- If *iFinger* or *iFingerAxis* is invalid a (cSDHErrorInvalidParameter*) exception is thrown.

Examples:

```
// Assuming 'hand' is a cSDH object ...

cout << "The 1st axis of finger 2 has real axis index " << hand.GetFingerNumberOfAxes( 2, 0 ) << "\n";
```

10.20.4.40 std::vector< double > cSDH::GetFingerEnable (std::vector< int > const & *fingers*) throw (cSDHLibraryException*)

Get enabled/disabled state of axis controllers of finger(s).

The enabled/disabled state of the controllers of the selected fingers is read from the SDH. A finger is reported disabled if any of its axes is disabled and reported enabled if all its axes are enabled.

Parameters

<i>fingers</i>	- A vector of finger indices to access.
----------------	---

- The indices in *fingers* are checked if they are valid finger indices.
- If **any** finger index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of enabled/disabled states as doubles (0=disabled, 1.0=enabled) of the selected axes.
- The order of the elements of the *axes* vector and the returned values vector *rv* matches. I.e. *rv*[*i*] will be the value of axis *axes*[*i*] (not axis *i*).

See also [GetAxisEnable\(int\)](#) for an overloaded variant to access a single axis.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get enabled state of all fingers:
std::vector<double> v = hand.GetFingerEnable( hand.all_fingers );
// now v is something like {0.0, 1.0, 0.0}

// Get enabled state of finger 0 and 2
std::vector<int> fingers02;
fingers02.push_back( 0 );
fingers02.push_back( 2 );

v = hand.GetFingerEnable( fingers02 );
// now v is something like {0.0, 0.0}

// Get enabled state of finger 1
double v1 = hand.GetFingerEnable( 1 );
// now v1 is something like 1.0
```

10.20.4.41 double cSDH::GetFingerEnable (int *iFinger*) throw (cSDHLibraryException*)

Like [GetFingerEnable\(std::vector<int>const&\)](#), just for a single finger *iFinger* and returning a single double value

10.20.4.42 std::vector< double > cSDH::GetFingerMaxAngle (int *iFinger*) throw (cSDHLibraryException*)

Get the maximum axis angles of a single finger.

The maximum axis angles of finger *iFingers* axes, stored in the library, are returned.

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index
----------------	--

Remarks

- The *iFinger* index is checked if it is a valid finger index.
- If *iFinger* is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A list of the selected fingers maximum axis angles
- The values are returned in the configured angle unit system [uc_angle](#).

See also [GetFingerMaxAngle\(int,double&,double&,double&\)](#) for an overloaded variant to get finger axis max angles into single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum axis angles of finger 0
std::vector<double> v = hand.GetFingerMaxAngle( 0 );
// now v is something like {90.0, 90.0, 90.0}

// Get target axis angles of finger 1
double a0, a1, a2;
hand.GetFingerMaxAngle( 1, a0, a1, a2 );
// now a0, a1, a2 are something like 90.0, 90.0, 90.0 respectively.

// Or if you change the angle unit system:
hand.UseRadians();
v = hand.GetFingerMaxAngle( 0 );
// now v is something like {1.5707963267948966, 1.5707963267948966, 1.5707963267948966}
```

10.20.4.43 void cSDH::GetFingerMaxAngle (int *iFinger*, double & *a0*, double & *a1*, double & *a2*) throw (cSDHLibraryException*)

Like [GetFingerMaxAngle\(int\)](#), just returning the finger axis max angles in the *a0*, *a1* and *a2* parameters which are given by reference.

10.20.4.44 void cSDH::GetFingerMinAngle (int *iFinger*, double & *a0*, double & *a1*, double & *a2*) throw (cSDHLibraryException*)

Like [GetFingerMinAngle\(int\)](#), just returning the finger axis min angles in the *a0*, *a1* and *a2* parameters which are given by reference.

10.20.4.45 std::vector< double > cSDH::GetFingerMinAngle (int *iFinger*) throw (cSDHLibraryException*)

Get the minimum axis angles of a single finger.

The minimum axis angles of finger *iFingers* axes, stored in the library, are returned.

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index
----------------	--

Remarks

- The *iFinger* index is checked if it is a valid finger index.
- If *iFinger* is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A list of the selected fingers minimum axis angles
- The values are returned in the configured angle unit system [uc_angle](#).

See also [GetFingerMinAngle\(int,double&,double&,double&\)](#) for an overloaded variant to get finger axis min angles into single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get minimum axis angles of finger 0
std::vector<double> v = hand.GetFingerMinAngle( 0 );
// now v is something like {0.0, -90.0, -90.0}

// Get target axis angles of finger 1
double a0, a1, a2;
hand.GetFingerMinAngle( 1, a0, a1, a2 );
// now a0, a1, a2 are something like 0.0, -90.0, -90.0 respectively.

// Or if you change the angle unit system:
hand.UseRadians();
v = hand.GetFingerMinAngle( 0 );
// now v is something like {0.0, -1.5707963267948966, -1.5707963267948966}
66}
```

10.20.4.46 int cSDH::GetFingerNumberOfAxes (int *iFinger*) throw (cSDHLibraryException*)

Return the number of real axes of finger with index *iFinger*.

Parameters

<i>iFinger</i>	- index of finger in range [0..NUMBER_OF_FINGERS-1]
----------------	---

Returns

- Number of real axes of finger with index *iFinger*
- If *iFinger* is invalid a (cSDHErrorInvalidParameter*) exception is thrown.

Examples:

```
// Assuming 'hand' is a cSDH object ...

cout << "The finger 0 has " << hand.GetFingerNumberOfAxes( 0 ) << " real
```

```
axes\n";
```

10.20.4.47 `std::vector< double > cSDH::GetFingerTargetAngle (int iFinger) throw (cSDHLibraryException*)`

Get the target axis angles of a single finger.

The target axis angles of finger *iFinger* are read from the SDH.

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index
----------------	--

Remarks

- The *iFinger* index is checked if it is a valid finger index.
- If *iFinger* is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A list of the selected fingers target axis angles
- The values are returned in the configured angle unit system [uc_angle](#).

See also [GetFingerTargetAngle\(int,double&,double&,double&\)](#) for an overloaded variant to get finger axis target angles into single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get target axis angles of finger 0
std::vector<double> v = hand.GetFingerTargetAngle( 0 );
// now v is something like {42.0, -10.0, 47.11}

// Get target axis angles of finger 1
double a0, a1, a2;
hand.GetFingerTargetAngle( 1, a0, a1, a2 );
// now a0, a1, a2 are something like 0.0, 24.7 and -5.5 respectively.
```

10.20.4.48 `void cSDH::GetFingerTargetAngle (int iFinger, double & a0, double & a1, double & a2) throw (cSDHLibraryException*)`

Like [GetFingerTargetAngle\(int\)](#), just returning the target axis angles in the *a0*, *a1* and *a2* parameters which are given by reference.

10.20.4.49 `std::vector< double > cSDH::GetFingerXYZ (int iFinger, double a0, double a1, double a2) throw (cSDHLibraryException*)`

Like [SetFingerTargetAngle\(int,std::vector<double>const&\)](#), just with individual finger axis angles *a0*, *a1* and *a2*.

10.20.4.50 `std::vector< double > cSDH::GetFingerXYZ (int iFinger, std::vector< double > const & angles) throw (cSDHLibraryException*)`

Get the cartesian xyz finger tip position of a single finger from the given axis angles (forward kinematics).

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index
<i>angles</i>	- a vector of NUMBER_OF_AXES_PER_FINGER angles. The values are expected in the configured angle unit system uc_angle .

Remarks

- The *iFinger* index is checked if it is a valid finger index.
- The angles are checked if they are in the allowed range [0 .. [f_max_angle_v](#)], i.e. it is checked that `angles[i]`, converted to internal units, is in [0 .. `f_max_angle_v[finger_axis_index[iFinger][i]]`].
- If any index or value is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

- A vector of the x,y,z values of the finger tip position
- The values are returned in the configured position unit system [uc_position](#).

See also [GetFingerXYZ\(int,double,double,double\)](#) for an overloaded variant to get finger tip position from single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get actual finger angles of finger 0:
std::vector<double> angles = hand.GetFingerActualAngle( 0 );

// Get actual finger tip position of finger 0:
std::vector<double> position = hand.GetFingerXYZ( 0, angles );
// now position is something like {18.821618775581801, 32.60000000000000
01, 174.0}
// (assuming that finger 0 is at axis angles {0,0,0})

// Get finger tip position of finger 2 at axis angles {90,-90,-90}:
position = hand.GetFingerXYZ( 2, 90, -90, -90 );
// now position is something like {18.821618775581804, 119.600000000000
02, -53.0}

// Or if you change the angle unit system:
hand.UseRadians();
position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.570796326794896
6, -1.5707963267948966 );
// now position is still something like {18.821618775581804, 119.600000
00000002, -53.0}

// Or if you change the position unit system too:
```

```

        hand.uc_position = &cSDH::uc_position_meter
        position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.570796326794896
6, -1.5707963267948966 );
        // now position is still something like {0.018821618775581, 0.119.60000
000000002, -0.0529999999999}

```

10.20.4.51 char const * cSDH::GetFirmwareRelease (void) throw (cSDHLibraryException*)

Return the actual release name of the firmware of the SDH (not the library) as string.

This will throw a (cSDHErrorCommunication*) exception if the connection to the SDH is not yet opened.

Examples:

```

// Assuming 'hand' is a cSDH object ...

cout << "The SDH firmware reports release " << hand.GetFirmwareRelease()
<< "\n";

```

See also

See [GetFirmwareReleaseRecommended\(\)](#) to get the recommended [SDH](#) firmware release.

10.20.4.52 char const * cSDH::GetFirmwareReleaseRecommended (void) [static]

Return the recommended release of the firmware of the SDH by this library as string.

Examples:

```

// static member function, so no cSDH object is needed for access:

cout << "This SDHLibrary recommends an SDH firmware release " <<
cSDH::GetFirmwareReleaseRecommended() << "\n";

```

See also

See [GetFirmwareRelease\(\)](#) to get the actual release of the [SDH](#) firmware

10.20.4.53 double cSDH::GetGripMaxVelocity (void)

Get the maximum velocity of grip skills

The maximum velocity is currently not read from the SDH, but is stored in the library.

Returns

- a single double value is returned representing the velocity in the [uc_angular_velocity](#) unit system

Examples:

```
// Assuming "hand" is a cSDH object ...

// Get maximum grip skill velocity
double v = hand.GetGripMaxVelocity();
// v is now something like 100.0

// Or if you change the velocity unit system:
hand.UseRadians();
v = hand.GetGripMaxVelocity();
// now v is something like 1.7453292519943295
```

10.20.4.54 char const * cSDH::GetInfo (char const * *what*) throw (cSDHLibraryException*)

Return info according to *what* # # The following values are valid for *what*: # - "date-library" : date of the SDHLibrary-python release # - "release-library" : release name of the sdh.py python module # - "release-firmware" : release name of the SDH firmware (requires # an opened communication to the SDH) # - "release-firmware-recommended" : recommended release name of the SDH # firmware # - "date-firmware" : date of the SDH firmware (requires # an opened communication to the SDH) # - "release-soc" : release name of the SDH SoC (requires # an opened communication to the SDH) # - "date-soc" : date of the SDH SoC (requires # an opened communication to the SDH) # - "id-sdh" : ID of SDH # - "sn-sdh" : Serial number of SDH # #

Examples:

```
#

# # Assuming 'hand' is a sdh.cSDH object ...
#
# print "The SDH firmware reports release %s" % ( hand.GetInfo( "release-f
#   irmware" ) )
#
#

# #
```

10.20.4.55 char const * cSDH::GetLibraryName (void) [static]

Return the name of the library as string.

Examples:

```
// static member function, so no cSDH object is needed for access:

cout << "The SDHLibrary reports name " << cSDH::GetLibraryName() << "\n"
;
```

10.20.4.56 `char const * cSDH::GetLibraryRelease (void) [static]`

Return the release name of the library (not the firmware of the SDH) as string.

Examples:

```
// static member function, so no cSDH object is needed for access:
cout << "The SDHLibrary reports release name " << cSDH::GetReleaseLibrary() << "\n";
```

10.20.4.57 `pSetFunction cSDH::GetMotorCurrentModeFunction (eMotorCurrentMode mode) throw (cSDHLibraryException*) [protected]`

Internal helper function: return the get/set function of the comm_interface object that is responsible for setting/getting motor currents in *mode*.

10.20.4.58 `std::vector< double > cSDH::GetTemperature (std::vector< int > const & sensors) throw (cSDHLibraryException*)`

Return temperature(s) measured within the SDH.

Parameters

<i>sensors</i>	- A vector of indices of temperature sensors to access. <ul style="list-style-type: none"> • index 0 is sensor near motor of axis 0 (root) • index 1 is sensor near motor of axis 1 (proximal finger 1) • index 2 is sensor near motor of axis 2 (distal finger 1) • index 3 is sensor near motor of axis 3 (proximal finger 2) • index 4 is sensor near motor of axis 4 (distal finger 2) • index 5 is sensor near motor of axis 5 (proximal finger 3) • index 6 is sensor near motor of axis 6 (distal finger 3) • index 7 is FPGA temperature (controller chip) • index 8 is PCB temperature (Printed Circuit Board)
----------------	--

Remarks

- The indices in *sensors* are checked if they are valid sensor indices.
- If **any** sensor index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

To access a single temperature sensor use [GetTemperature\(int\)](#), see there.

Returns

The temperatures of the selected sensors are returned as `std::vector<double>` in the configured temperature unit system [uc_temperature](#).

Examples:

```

// Assuming 'hand' is a cSDH object ...

// Get measured values of all sensors
std::vector<double> temps = hand.GetTemperature( hand.all_temperature_sensors );
// Now temps is something like { 38.500,37.250,35.750,37.250,33.500,36.500,32.250,59.625,52.500 }

// Get controller temperature only:
double temp_controller = hand.GetTemperature( 0 );
// Now temp_controller is something like 40.5

// If we - for some obscure islandish reason - would want
// temperatures reported in degrees fahrenheit, the unit
// converter can be changed:
hand.uc_temperature = &cSDH::uc_temperature_fahrenheit;

// Get all temperatures again:
temps = hand.GetTemperature( hand.all_temperature_sensors );
// Now temps is something like {100.0, 96.8, 92.3, 97.7, 91.8, 96.8, 90.1, 137.5, 125.2}

```

10.20.4.59 double cSDH::GetTemperature (int *iSensor*) throw (cSDHLibraryException*)

Like [GetTemperature\(std::vector<int>const&\)](#), just for one sensor *iSensor* and returning a single temperature as double.

10.20.4.60 cSDHBase::eVelocityProfile cSDH::GetVelocityProfile (void) throw (cSDHLibraryException*)

Get the type of velocity profile used in the SDH

Returns

the currently set velocity profile as integer, see [eVelocityProfileType](#)

Examples:

```

// Assuming 'hand' is a cSDH object ...

// Get the velocity profile from the SDH:
velocity_profile = hand.GetVelocityProfile();
// now velocity_profile is something like eVP_SIN_SQUARE or eVP_RAMP

```

10.20.4.61 double cSDH::GripHand (eGraspId *grip*, double *close*, double *velocity*, bool *sequence=true*) throw (cSDHLibraryException*)

Perform one of the internal [eGraspId](#) "grips" or "grasps"

Warning

THIS DOES NOT WORK WITH SDH FIRMWARE PRIOR TO 0.0.2.6 AND SDHLIBRARY-CPP PRIOR TO 0.0.1.12 This was a feature in the ancient times of the SDH1 and now does work again for SDH firmware 0.0.2.6 and newer and SDHLIBRARY-CPP 0.0.1.12 and newer. We intend to further improve this feature (e.g. store user defined grips within the SDH) in the future, but a particular deadline has not been determined yet.

Bug

With SDH firmware > 0.0.2.6 and SDHLibrary < 0.0.1.12 [GripHand\(\)](#) does not work ([Bug 575](#))
=> **Resolved in SDHLibrary 0.0.1.12**

Bug

With SDH firmware < 0.0.2.6 [GripHand\(\)](#) does not work and might yield undefined behaviour there
=> **Resolved in SDH firmware 0.0.2.6**

Bug

Currently the performing of a skill or grip with [GripHand\(\)](#) can **NOT** be interrupted!!! Even if the command is sent with *sequ=false* it **cannot** be stopped or emergency stopped.

Parameters

<i>grip</i>	- The index of the grip to perform [0..eGID_DIMENSION-1] (s.a. eGraspId)
<i>close</i>	- close-ratio: [0.0 .. 1.0] where 0.0 is 'fully opened' and 1.0 is 'fully closed'
<i>velocity</i>	- maximum allowed angular axis velocity in the chosen external unit system uc_angular_velocity
<i>sequ</i>	- flag: if true (default) then the function executes sequentially and returns not until after the SDH has finished the movement. If false then the function returns immediately after the movement command has been sent to the SDH.

- The *close* and *velocity* values are checked if they are in their allowed range.
- If **any** value is invalid then **no** grip is performed, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

The expected/elapsed execution time for the movement in the configured time unit system [uc_time](#).

Remarks

- Currently the actual movement velocity of an axis is determined by the SDH firmware to make the movements of all involved axes start and end syn-

chronously at the same time. Therefore the axis that needs the longest time for its movement at its given maximum velocity determines the velocities of all the other axes.

- The currently set target axis angles are not changed by this command
- The movement uses the eMotorCurrentMode motor current modes "eMCM_GRIP" while gripping and then changes the motor current mode to "eMCM_HOLD". After the movement previously set motor currents set for mode "eMCM_MOVE" are **overwritten**!

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Perform a fully opened central grip at 50 degrees per second:
hand.GripHand( hand.eGID_CENTRICAL, 0.0, 50.0, true );

// Now close it 50% with 30 degrees per second:
hand.GripHand( hand.eGID_CENTRICAL, 0.5, 30.0, true );

// Then close it completely with 20 degrees per second:
hand.GripHand( hand.eGID_CENTRICAL, 1.0, 20.0, true );
```

10.20.4.62 bool cSDH::IsOpen (void) throw () [virtual]

Return true if connection to SDH firmware/hardware is open.

Implements [SDH::cSDHBase](#).

10.20.4.63 bool cSDH::IsVirtualAxis (int iAxis) throw (cSDHLibraryException*)

Return true if index *iAxis* refers to a virtual axis.

10.20.4.64 double cSDH::MoveAxis (std::vector< int >const & axes, bool sequ = true) throw (cSDHLibraryException*)

Move selected axis/axes to the previously set target pose with the previously set velocity profile, (maximum) target velocities and target accelerations

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>sequ</i>	- flag: if true (default) then the function executes sequentially and returns not until after the SDH has finished the movement. If false then the function returns immediately after the movement command has been sent to the SDH (the currently set target axis angles for other axes will then be overwritten with their current actual axis angles).

- The indices in *axes* are checked if they are valid axis indices.

- If any index is invalid then no movement is performed, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

The expected/elapsed execution time for the movement in the configured time unit system [uc_time](#)

Remarks

- The axes will be enabled automatically.
- Currently the actual movement velocity of an axis is determined by the SDH firmware to make the movements of all involved axes start and end synchronously at the same time. Therefore the axis that needs the longest time for its movement at its given maximum velocity determines the velocities of all the other axes.
- Other axes than those selected by *axes* will **NOT** move, even if target axis angles for the axes have been set. (Remember: as axis 0 is used by finger 0 and 2 these two fingers cannot be moved completely independent of each other.)
- If *sequ* is true then the currently set target axis angles for other fingers will be restored upon return of the function.
- If *sequ* is false then the currently set target axis angles for other fingers will be **overwritten** with their current actual axis angles

See also [MoveAxis\(int,bool\)](#) for an overloaded variant to move a single axis.

Examples:

```
// Assuming 'hand' is a cSDH object ...

// create an index vector for adressing axes 0, 4 and 2 (in that order)

std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );

// Set a new target pose for axes 0, 4 and 2:
std::vector<double> angles042;
angles042.push_back( 0.0 );
angles042.push_back( -44.4 );
angles042.push_back( -22.2 );

hand.SetFingerTargetAngle( axes042, angles042 );

// First move Axis 0 only to its new target position:
hand.MoveAxis( 0 );

// The axis 0 has now reached its target position 0.0 degrees. The
// target poses for axes 4 and 2 are still set since the
// last MoveAxes() call was sequentially (und thus it could
// restore the previously set target axis angles of not
```

```

// selected axes after the movement finished)

// So move axes 4 and 2 now, this time non-sequentially:
std::vector<int> axes42;
axes42.push_back( 4 );
axes42.push_back( 2 );

double t = hand.MoveAxes( axis42, false );

// The two axes 4 and 2 are now moving to their target position.
// We have to wait until the non-sequential call has finished:
SleepSec( t );

// The axes 4 and 2 have now moved to -44.4 and -22.2.

// The target angles for other axes have by now been
// overwritten since the last MoveAxis() call was
// non-sequentially (und thus it could \b NOT restore the
// previously set target axis angles of not selected axes
// after the movement finished)

// Set new target angles for all axes ("home pose");
hand.SetAxisTargetAngle( hand.All, 0.0 );

// Now move all axes back to home pose:
hand.MoveAxes( hand.All );

```

Bug

With SDH firmware < 0.0.2.7 calling [MoveAxis\(\)](#) while some axes are moving in eCT_POSE controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

10.20.4.65 `double cSDH::MoveAxis (int iAxis, bool sequ = true) throw (cSDHLibraryException*)`

Like [MoveAxis\(std::vector<int>const&,bool\)](#), just for a single axis *iAxis* (or all axes if [All](#) is given).

10.20.4.66 `double cSDH::MoveFinger (int iFinger, bool sequ = true) throw (cSDHLibraryException*)`

Like [MoveFinger\(std::vector<int>const&,bool\)](#), just for a single finger *iFinger* (or all fingers if [All](#) is given).

10.20.4.67 `double cSDH::MoveFinger (std::vector< int >const & fingers, bool sequ = true)
throw (cSDHLibraryException*)`

Move selected finger(s) to the previously set target pose with the previously set velocity profile, (maximum) target velocities and target accelerations.

Parameters

<i>fingers</i>	- A vector of finger indices to access.
<i>sequ</i>	- flag: if true (default) then the function executes sequentially and returns not until after the SDH has finished the movement. If false then the function returns immediately after the movement command has been sent to the SDH (the currently set target axis angles for other fingers will then be overwritten with their current actual axis angles).

- The indices in *fingers* are checked if they are valid finger indices.
- If any index is invalid then no movement is performed, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Returns

The expected/elapsed execution time for the movement in the configured time unit system [uc_time](#)

Remarks

- The axes will be enabled automatically.
- Currently the actual movement velocity of an axis is determined by the SDH firmware to make the movements of all involved axes start and end synchronously at the same time. Therefore the axis that needs the longest time for its movement at its given maximum velocity determines the velocities of all the other axes.
- Other fingers than *iFinger* will **NOT** move, even if target axis angles for their axes have been set. (Exception: as axis 0 is used by finger 0 and 2 these two fingers cannot be moved completely independent of each other.)
- If *sequ* is true then the currently set target axis angles for other fingers will be restored upon return of the function.
- If *sequ* is false then the currently set target axis angles for other fingers will be **overwritten** with their current actual axis angles

See also [MoveFinger\(int,bool\)](#) for an overloaded variant to move a single finger.

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Set a new target pose for finger 0:
hand.SetFingerTargetAngle( 0, 0.0, 0.0, 0.0 );

// Set a new target pose for finger 1
```

```

hand.SetFingerTargetAngle( 1, 0.0, -10.0, -10.0 );

// Set a new target pose for finger 2
hand.SetFingerTargetAngle( 2, 20.0, -20.0, -20.0 );

// Move finger 0 only (and finger 2 partly as axis 0 also belongs to fi
nger 2);
hand.MoveFinger( 0, true );
// The finger 0 has been moved to {20,0,0}
// (axis 0 is 'wrong' since the target angle for axis 0 has been overwr
itten
// while setting the target angles for finger 2);

// The target poses for finger 1 and 2 are still set since the
// last MoveFinger() call was sequentially.
// So move finger 1 now:
double t = hand.MoveFinger( 1, false );

// wait until the non-sequential call has finished:
SleepSec( t );

// The finger 1 has been moved to {0,-10,-10}.

// The target angles for finger 2 have been overwritten since the
// last MoveFinger() call was non-sequentially.

// Therefore this next call will just keep the fingers in their
// current positions:
hand.MoveFinger( hand.All, true );

// Set new target angles for all axes ("home pose");
hand.SetAxisTargetAngle( hand.All, 0.0 );

// Now move all axes back to home pose:
hand.MoveHand();

```

Bug

With SDH firmware < 0.0.2.7 calling [MoveFinger\(\)](#) while some axes are moving in eCT_POSE controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

10.20.4.68 `double cSDH::MoveHand(bool sequ = true) throw (cSDHLibraryException*)`

Move all fingers to the previously set target pose with the previously set (maximum) velocities.

This is just a shortcut to [MoveFinger\(int,bool\)](#) with *iFinger* set to `hand.All` and *sequ* as indicated, so see there for details and examples.

Bug

With SDH firmware < 0.0.2.7 calling [MoveHand\(\)](#) while some axes are moving in eCT_POSE controller type will make the joints jerk. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

10.20.4.69 `void cSDH::OpenCAN.ESD (int _net = 0, unsigned long _baudrate = 1000000, double _timeout = 0.0, Int32 _id_read = 43, Int32 _id_write = 42) throw (cSDHLibraryException*)`

Open connection to SDH via CAN using an ESD CAN card. If the library was compiled without ESD CAN support then this will just throw an exception. See setting for WITH_ESD_CAN in the top level makefile.

Parameters

<code>_net</code>	: The ESD CAN net number of the CAN port to use. (default: 0)
<code>_baudrate</code>	: the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default), 800000, 500000, 250000, 125000, 100000, 50000, 20000, 10000)
<code>_timeout</code>	: The timeout to use: <ul style="list-style-type: none"> • <code><= 0</code> : wait forever (default) • <code>T</code> : wait for T seconds
<code>_id_read</code>	- the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)
<code>_id_write</code>	- the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x02a)

Examples:

```
// Assuming 'hand' is a cSDH object ...

// use default parameters for net, baudrate, timeout and IDs
hand.OpenCAN_ESD ( );

// use non default settings:
// net=1, baudrate=500000, timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_ESD( 1, 500000, 1.0, 0x143, 0x142 );
```

10.20.4.70 `void cSDH::OpenCAN.ESD (tDeviceHandle _ntcan_handle, double _timeout = 0.0, Int32 _id_read = 43, Int32 _id_write = 42) throw (cSDHLibraryException*)`

Open connection to SDH via CAN using an ESD CAN card using an existing handle. If the library was compiled without ESD CAN support then this will just throw an exception. See setting for WITH_ESD_CAN in the top level makefile.

Parameters

<code>_ntcan_handle</code>	: the ESD CAN handle to reuse (please cast your NTCAN_HANDLE to tDeviceHandle! It is save to do that!)
<code>_timeout</code>	: The timeout to use: <ul style="list-style-type: none"> • <code><= 0</code> : wait forever (default) • <code>T</code> : wait for T seconds
<code>_id_read</code>	- the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)
<code>_id_write</code>	- the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x2a)

Examples:

```
// Assuming 'hand' is a cSDH object ...
// and 'handle' is a valid NTCAN_HANDLE for the ESD device

// use default parameters for timeout and IDs
hand.OpenCAN_ESD( tDeviceHandle(handle) );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_ESD( handle, 1.0, 0x143, 0x142 );
```

10.20.4.71 `void cSDH::OpenCAN_PEAK (unsigned long _baudrate = 1000000, double _timeout = 0.0, Int32 _id_read = 43, Int32 _id_write = 42, const char * _device = "/dev/pcanusb0") throw (cSDHLibraryException*)`

Open connection to SDH via CAN using an PEAK CAN card. If the library was compiled without PEAK CAN support then this will just throw an exception. See setting for WITH_PEAK_CAN in the top level makefile.

Parameters

<code>_baudrate</code>	: the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default),800000,500000,250000,125000,100000,50000,20000,10000)
<code>_timeout</code>	: The timeout to use: <ul style="list-style-type: none"> • <code><= 0</code> : wait forever (default) • <code>T</code> : wait for T seconds
<code>_id_read</code>	- the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)
<code>_id_write</code>	- the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x02a)
<code>_device</code>	- the PEAK device name. Used for the Linux char dev driver only. default="/dev/pcanusb0"

Examples:

```
// Assuming 'hand' is a cSDH object ...
```



```
// use default parameters for baudrate, timeout, IDs and device
hand.OpenCAN_PEAK( );

// use non default settings:
// baudrate=500000, timeout=1.0, id_read=0x143, id_write=0x142, , const
char *device="/dev/pcanusb1"
hand.OpenCAN_PEAK( 500000, 1.0, 0x143, 0x142, "/dev/pcanusb1" );
```

10.20.4.72 void **cSDH::OpenCAN_PEAK** (tDeviceHandle *_handle*, double *_timeout* = 0.0, Int32 *_id_read* = 43, Int32 *_id_write* = 42) throw (cSDHLibraryException*)

Open connection to SDH via CAN using an PEAK CAN card using an existing handle. If the library was compiled without PEAK CAN support then this will just throw an exception. See setting for WITH_PEAK_CAN in the top level makefile.

Parameters

<i>_handle</i>	: The PEAK CAN handle to reuse to connect to the PEAK CAN driver (please cast your PEAK_HANDLE to tDeviceHandle! It is save to do that!)
<i>_timeout</i>	: The timeout to use: <ul style="list-style-type: none"> • <= 0 : wait forever (default) • T : wait for T seconds
<i>_id_read</i>	- the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)
<i>_id_write</i>	- the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x2a)

Examples:

```
// Assuming 'hand' is a cSDH object ...
// and 'handle' is a valid HANDLE for the PEAK device

// use default parameters for timeout and IDs
hand.OpenCAN_PEAK( tDeviceHandle(handle) );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_PEAK( handle, 1.0, 0x143, 0x142 );
```

10.20.4.73 void **cSDH::OpenRS232** (int *_port* = 0, unsigned long *_baudrate* = 115200, double *_timeout* = -1, char const * *_device_format_string* = "/dev/ttyS%d") throw (cSDHLibraryException*)

Open connection to SDH via RS232.

Parameters

<i>_port</i>	: The number of the serial port to use. The default value port=0 refers to 'COM1' in Windows and to the corresponding '/dev/ttyS0' in Linux.
--------------	--

<i>_baudrate_</i> :	the baudrate in bit/s, the default is 115200 which happens to be the default for the SDH too
<i>_timeout</i>	: The timeout to use: <ul style="list-style-type: none"> • -1 : wait forever • T : wait for T seconds
<i>_device_format_string</i>	: a format string (C string) for generating the device name, like "/dev/ttyS%d" (default) or "/dev/ttyUSB%d". Must contain a d where the port number should be inserted. This char array is duplicated on construction. When compiled with VCC (MS-Visual C++) then this is not used.

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Open connection to SDH via default port:
hand.OpenRS232();

// Use a different port 2 == COM3 == /dev/ttyS2 for a second hand "hand2":
cSDH hand2();
hand2.OpenRS232( 2 );

// Linux only: Use a different USB to RS232 device on port 3 /dev/ttyUSB
3 for a third hand "hand3":
cSDH hand3();
hand2.OpenRS232( 3, 115200, -1, "/dev/ttyUSB%d" );
```

10.20.4.74 `void cSDH::OpenTCP (char const * _tcp_adr = "192.168.1.1", int _tcp_port = 23, double _timeout = 0.0) throw (cSDHLibraryException*)`

Open connection to SDH via TCP using TCP/IP address *_tcp_adr* and *_tcp_port*.

Parameters

<i>_tcp_adr</i>	: The tcp host address of the SDH . Either a numeric IP as string or a host-name
<i>_tcp_port</i>	: the tcp port number on the SDH to connect to
<i>_timeout</i>	: The timeout to use: <ul style="list-style-type: none"> • <= 0 : wait forever (default) • T : wait for T seconds

Examples:

```
// Assuming 'hand' is a cSDH object ...

// IP-address 192.168.1.1, port 23, timeout=0.0
hand.OpenTCP( "192.168.1.1.", 23, 0.0 );
```

10.20.4.75 `void cSDH::SetAxisEnable (std::vector< int > const & axes, std::vector< bool > const & states) throw (cSDHLibraryException*)`

Like [SetAxisEnable\(std::vector<int>const&,std::vector<double>const&\)](#), just accepting a vector of `bool` values as states, see there.

10.20.4.76 `void cSDH::SetAxisEnable (std::vector< int > const & axes, std::vector< double > const & states) throw (cSDHLibraryException*)`

Set enabled/disabled state of axis controller(s).

The controllers of the selected axes are enabled/disabled in the SDH. Disabled axes are not powered and thus might not remain in their current pose due to gravity, inertia or other external influences. But to prevent overheating the axis controllers should be switched of when not needed.

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>states</i>	- A vector of enabled states (0 = disabled, !=0 = enabled) to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set enabled state will be kept for the corresponding axis.

Remarks

- The lengths of the *axes* and *states* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `state[i]` will be applied to axis `axes[i]` (not axis `i`).
- The indices are checked if they are valid axis indices.
- If **any** index is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

See also [SetAxisEnable\(int,double\)](#), [SetAxisEnable\(int,bool\)](#) for overloaded variants to set a single axis enabled/disabled or to set the same state for all axes. See further [SetAxisEnable\(std::vector<int>const&,std::vector<bool>const&\)](#) for a variant that accepts a `bool` vector for the states to set.

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Enable all axes:
hand.SetAxisEnable( hand.all_axes, hand.ones_v );

// Disable all axes:
hand.SetAxisEnable( All, 0 );

// Enable axis 0 and 2 while disabling axis 4:
std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
```

```

std::vector<double> states042;
states042.push_back( 1.0 );
states042.push_back( 0.0 );
states042.push_back( 1.0 );

hand.SetAxisEnable( axes042, states042 );

// Disable axis 2
hand.SetAxisEnable( 2, false );

```

10.20.4.77 void `cSDH::SetAxisEnable (int iAxis = All, double state = 1.0) throw (cSDHLibraryException*)`

Like `SetAxisEnable(std::vector<int>const&,std::vector<double>const&)`, just for a single axis *iAxis* and a single axis state *state*, see there.

If *iAxis* is [All](#) then *state* is applied to all axes.

10.20.4.78 void `cSDH::SetAxisEnable (int iAxis = All, bool state = true) throw (cSDHLibraryException*)`

Like `SetAxisEnable(std::vector<int>const&,std::vector<double>const&)`, just for a single axis *iAxis* and a single axis state *state*, see there.

If *iAxis* is [All](#) then *state* is applied to all axes.

10.20.4.79 void `cSDH::SetAxisMotorCurrent (int iAxis, double motor_current, eMotorCurrentMode mode = eMCM_MOVE) throw (cSDHLibraryException*)`

Like `SetAxisMotorCurrent(std::vector<int>const&,std::vector<double>const&,eMotorCurrentMode)`, just for a single axis *iAxis* and a single motor current *motor_current*, see there.

If *iAxis* is [All](#) then *motor_current* is set for all axes.

10.20.4.80 void `cSDH::SetAxisMotorCurrent (std::vector< int > const & axes, std::vector< double > const & motor_currents, eMotorCurrentMode mode = eMCM_MOVE) throw (cSDHLibraryException*)`

Set the maximum allowed motor current(s) for axes.

The maximum allowed motor currents are sent to the SDH. The motor currents can be stored:

- axis specific
- mode specific (see [eMotorCurrentMode](#))

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>motor_</i> - <i>currents</i>	- A vector of motor currents to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis motor current will be kept for the corresponding axis. The value(s) are expected in the configured motor current unit system uc_motor_current .
<i>mode</i>	- the mode to set the maximum motor current for. One of the eMotorCurrentMode modes.

Remarks

- The lengths of the *axes* and *motor_currents* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `motor_currents[i]` will be applied to axis `axes[i]` (not axis `i`).
- The indices are checked if they are valid axis indices.
- The motor currents are checked if they are in the allowed range `[0 .. f_max_motor_current_v]`, i.e. it is checked that `motor_currents[i]`, converted to internal units, is in `[0 .. f_max_motor_currents_v[axes[i]]]`.
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter](#)* exception is thrown.

See also [SetAxisMotorCurrent\(int,double,eMotorCurrentMode\)](#) for an overloaded variant to set a single axis motor current or to set the same motor current for all axes.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set maximum allowed motor current of all axes to the given values in
mode "eMCM_MOVE":
std::vector<double> all_motor_currents;
all_motor_currents.push_back( 0.0 );
all_motor_currents.push_back( 0.1 );
all_motor_currents.push_back( 0.2 );
all_motor_currents.push_back( 0.3 );
all_motor_currents.push_back( 0.4 );
all_motor_currents.push_back( 0.5 );
all_motor_currents.push_back( 0.6 );

hand.SetAxisMotorCurrent( hand.all_axes, all_motor_currents );

// Set maximum allowed motor current of all axes to 0.1 A in mode "eMCM_
_HOLD":
hand.SetAxisMotorCurrent( hand.All, 1.0, eMCM_HOLD );

// Set maximum allowed motor current of axis 3 to 0.75 A in mode "eMCM_
_MOVE":
hand.SetAxisMotorCurrent( 3, 0.75, eMCM_MOVE );

// Set maximum allowed motor current of for axis 0, 4 and 2 to 0.0 A,
// 0.4 A and 0.2 A respectively in mode "eMCM_GRIP"
std::vector<int> axes042;
```

```

axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> motor_currents042;
motor_currents042.push_back( 0.0 );
motor_currents042.push_back( 0.4 );
motor_currents042.push_back( 0.2 );

hand.SetAxisMotorCurrent( axes042, states042, eMCM_GRIP );

```

10.20.4.81 void cSDH::SetAxisTargetAcceleration (int *iAxis*, double *acceleration*) throw (cSDHLibraryException*)

Like [SetAxisTargetAcceleration\(std::vector<int>const&,std::vector<double>const&\)](#), just for a single axis *iAxis* and a single acceleration *acceleration*, see there for details and examples.

10.20.4.82 void cSDH::SetAxisTargetAcceleration (std::vector< int >const & *axes*, std::vector< double >const & *accelerations*) throw (cSDHLibraryException*)

Set the target acceleration(s) for axis(*axes*).

The target accelerations are stored in the SDH and are used only for:

- the eCT_POSE controller type with eVP_RAMP velocity profile
- the eCT_VELOCITY_ACCELERATION controller type

Setting the target acceleration will not affect an ongoing movement, nor will it start a new movement. To take effect an additional command must be sent:

- in eCT_POSE controller type a move command: [MoveAxis\(\)](#) [MoveFinger\(\)](#) [MoveHand\(\)](#)
- in eCT_VELOCITY_ACCELERATION controller type the velocity must be set: [SetAxisTargetVelocity\(\)](#)

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>accelerations</i>	- A vector of axis target accelerations to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis target angle will be kept for the corresponding axis. The value(s) are expected in the configured angular acceleration unit system uc_angular_acceleration .

Remarks

- The lengths of the *axes* and *accelerations* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `accelerations[i]` will be applied to axis `axes[i]` (not axis `i`).

- The indices are checked if they are valid axis indices.
- The accelerations are checked if they are in the allowed range [0 .. [f_max_velocity_v](#)], i.e. it is checked that `accelerations[i]`, converted to internal units, is in [0 .. `f_max_velocity_v[axes[i]]`].
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

See also [SetAxisTargetAcceleration\(int,double\)](#) for an overloaded variant to set a single axis target acceleration or to set the same target acceleration for all axes.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set target axis acceleration of all axes to the given values:
std::vector<double> all_accelerations;
all_accelerations.push_back( 100.0 );
all_accelerations.push_back( 101.0 );
all_accelerations.push_back( 102.0 );
all_accelerations.push_back( 103.0 );
all_accelerations.push_back( 104.0 );
all_accelerations.push_back( 105.0 );
all_accelerations.push_back( 106.0 );

hand.SetAxisTargetAcceleration( hand.all_axes, all_accelerations );

// Set target axis acceleration of axis 3 to 420 degrees per square-second:
hand.SetAxisTargetAcceleration( 3, 420.0 );

// Set target acceleration of for axis 0,4 and 2 to 0.0, 444.0 and 222
degrees per square-second respectively:
std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> accelerations042;
accelerations042.push_back( 100.0 );
accelerations042.push_back( 104.0 );
accelerations042.push_back( 102.0 );

hand.SetAxisTargetAcceleration( axes042, accelerations042 );

// Set target axis acceleration of all axes to 142.1 degrees per square-second
hand.SetAxisTargetAcceleration( hand.All, 142.1 );
```

10.20.4.83 void cSDH::SetAxisTargetAngle (int *iAxis*, double *angle*) throw (cSDHLibraryException*)

Like [SetAxisTargetAngle\(std::vector<int>const&,std::vector<double>const&\)](#), just for a single axis *iAxis* and a single angle *angle*, see there for details and examples.

If *iAxis* is [All](#) then *motor_current* is set for all axes.

10.20.4.84 `void cSDH::SetAxisTargetAngle (std::vector< int > const & axes, std::vector< double > const & angles) throw (cSDHLibraryException*)`

Set the target angle(s) for axis(axes).

The target angles are stored in the SDH, the movement is not executed until an additional move command is sent.

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>angles</i>	- A vector of axis target angles to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis target angle will be kept for the corresponding axis. The value(s) are expected in the configured angle unit system uc_angle .

Remarks

- Setting the target angle will **not** make the axis/axes move.
- The lengths of the *axes* and *angles* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `angles[i]` will be applied to axis `axes[i]` (not axis `i`).
- The indices are checked if they are valid axis indices.
- The angles are checked if they are in the allowed range [[f_min_angle_v](#) .. [f_max_angle_v](#)], i.e. it is checked that `angles[i]`, converted to internal units, is in [[f_min_angle_v](#)[`axes[i]`] .. [f_max_angle_v](#)[`axes[i]`]].
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

See also [SetAxisTargetAngle\(int,double\)](#) for an overloaded variant to set a single axis target angle or to set the same target angle for all axes.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set target axis angle of all axes to the given values:
std::vector<double> all_angles;
all_angles.push_back( 0.0 );
all_angles.push_back( -11.0 );
all_angles.push_back( -22.0 );
all_angles.push_back( -33.0 );
all_angles.push_back( -44.0 );
all_angles.push_back( -55.0 );
all_angles.push_back( -66.0 );

hand.SetAxisTargetAngle( hand.all_axes, all_angles );

// Set target axis angle of axis 3 to -42 degrees:
hand.SetAxisTargetAngle( 3, -42.0 );

// Set target angle of for axis 0, 4 and 2 to 0.0, -44.4 and -2.22 degrees respectively:
```



```

std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> angles042;
angles042.push_back( 0.0 );
angles042.push_back( -44.4 );
angles042.push_back( -2.22 );

hand.SetAxisTargetAngle( axes042, angles042 );

// Set target axis angle of all axes to 0 degrees (home-position)
hand.SetAxisTargetAngle( hand.All, 0.0 );

```

10.20.4.85 std::vector< double > cSDH::SetAxisTargetGetAxisActualAngle (std::vector< int > const & axes, std::vector< double > const & angles) throw (cSDHLibraryException*)

Set the **target** angle(s) and get the **actual** angle(s) for axis(axes).

Opposed to [SetAxisTargetAngle\(\)](#) this will make the fingers move to the set target angles immediately, if the axis controllers are already enabled!

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>angles</i>	- A vector of axis target angles to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis target angle will be kept for the corresponding axis. The value(s) are expected in the configured angle unit system uc_angle .

Returns

the actual angle(s) of the selected axes

Remarks

- The lengths of the *axes* and *angles* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `angles[i]` will be applied to axis `axes[i]` (not axis `i`).
- The indices are checked if they are valid axis indices.
- The angles are checked if they are in the allowed range [`f_min_angle_v` .. `f_max_angle_v`], i.e. it is checked that `angles[i]`, converted to internal units, is in [`f_min_angle_v[axes[i]]` .. `f_max_angle_v[axes[i]]`].
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Examples:

```
// Assuming "hand" is a cSDH object ...
```

```

// Set target axis angles of all axes to the given values and read back
the actual angle:
std::vector<double> all_target_angles;
all_target_angles.push_back( 0.0 );
all_target_angles.push_back( 11.0 );
all_target_angles.push_back( 22.0 );
all_target_angles.push_back( 33.0 );
all_target_angles.push_back( 44.0 );
all_target_angles.push_back( 55.0 );
all_target_angles.push_back( 66.0 );

std::vector<double> all_acutal_angles;
all_acutal_angles = hand.SetAxisTargetGetAxisActualAngle( hand.all_axes
, all_target_angles );

// Set target angle of for axis 0,4 and 2 to 0.0, 44.4 and 2.22 degrees
per second respectively:
std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> target_angles042;
target_angles042.push_back( 0.0 );
target_angles042.push_back( 44.4 );
target_angles042.push_back( 2.22 );

std::vector<double> actual_angles042;
actual_angles042 = hand.SetAxisTargetGetAxisActualAngle( axes042, targe
t_angles042 );

```

10.20.4.86 `std::vector< double > cSDH::SetAxisTargetGetAxisActualVelocity (std::vector< int > const & axes, std::vector< double > const & velocities) throw (cSDHLibraryException*)`

Set the **target** velocity(s) and get the **actual** velocity(s) for axis(axes).

The target velocities are stored in the SDH. The time at which a new target velocities will take effect depends on the current axis controller type:

- in eCT_POSE controller type the new target velocities will not take effect until an additional move command is sent: [MoveAxis\(\)](#), [MoveFinger\(\)](#), [MoveHand\(\)](#)
- in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller type the new target velocity will take effect immediately, if the axis controllers are already enabled. This means that in eCT_VELOCITY_ACCELERATION controller type the accelerations must be set with [SetAxisTargetAcceleration\(\)](#) **before** calling [SetAxisTargetVelocity\(\)](#).

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>velocities</i>	- A vector of axis target velocities to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis target velocity will be kept for the corresponding axis. The value(s) are expected in the configured angular velocity unit system uc_angular_velocity .

Returns

the actual velocity(s) of the selected axes

Remarks

- The lengths of the *axes* and *velocities* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `velocities[i]` will be applied to axis `axes[i]` (not axis `i`).
- The indices are checked if they are valid axis indices.
- The velocities are checked if they are in the allowed range:
 - in `eCT_POSE` controller type: `[0 .. f_max_velocity_v]`, i.e. it is checked that `velocities[i]`, converted to internal units, is in `[0 .. f_max_velocity_v[axes[i]]]`.
 - in `eCT_VELOCITY` and `eCT_VELOCITY_ACCELERATION` controller type: `[-f_max_velocity_v .. f_max_velocity_v]`, i.e. it is checked that `velocities[i]`, converted to internal units, is in `[-f_max_velocity_v[axes[i]] .. f_max_velocity_v[axes[i]]]`.
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a `SDH::cSDHErrorInvalidParameter*` exception is thrown.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set target axis velocity of all axes to the given values and read back the actual velocity:
std::vector<double> all_target_velocities;
all_target_velocities.push_back( 0.0 );
all_target_velocities.push_back( 11.0 );
all_target_velocities.push_back( 22.0 );
all_target_velocities.push_back( 33.0 );
all_target_velocities.push_back( 44.0 );
all_target_velocities.push_back( 55.0 );
all_target_velocities.push_back( 66.0 );

std::vector<double> all_acutal_velocities;
all_acutal_velocities = hand.SetAxisTargetGetAxisActualVelocity( hand.all_axes, all_target_velocities );

// Set target velocity of for axis 0,4 and 2 to 0.0, 44.4 and 2.22 degrees per second respectively:
std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> target_velocities042;
target_velocities042.push_back( 0.0 );
target_velocities042.push_back( 44.4 );
target_velocities042.push_back( 2.22 );

std::vector<double> actual_velocities042;
actual_velocities042 = hand.SetAxisTargetGetAxisActualVelocity( axes042, target_velocities042 );
```

10.20.4.87 void `cSDH::SetAxisTargetVelocity (int iAxis, double velocity) throw (cSDHLibraryException*)`

Like `SetAxisTargetVelocity(std::vector<int>const&,std::vector<double>const&)`, just for a single axis *iAxis* and a single velocity *velocity*, see there for details and examples.

10.20.4.88 void `cSDH::SetAxisTargetVelocity (std::vector< int > const & axes, std::vector< double > const & velocities) throw (cSDHLibraryException*)`

Set the target velocity(s) for axis(axes).

The target velocities are stored in the SDH. The time at which a new target velocities will take effect depends on the current axis controller type:

- in eCT_POSE controller type the new target velocities will not take effect until an additional move command is sent: `MoveAxis()`, `MoveFinger()`, `MoveHand()`
- in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller type the new target velocity will take effect immediately, if the axis controllers are already enabled. This means that in eCT_VELOCITY_ACCELERATION controller type the accelerations must be set with `SetAxisTargetAcceleration()` before calling `SetAxisTargetVelocity()`.

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>velocities</i>	- A vector of axis target angles to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set axis target velocity will be kept for the corresponding axis. The value(s) are expected in the configured angular velocity unit system <code>uc_angular_velocity</code> .

Remarks

- The lengths of the *axes* and *velocities* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. *velocities*[i] will be applied to axis *axes*[i] (not axis *i*).
- The indices are checked if they are valid axis indices.
- The velocities are checked if they are in the allowed range:
 - in eCT_POSE controller type: [0 .. `f_max_velocity_v`], i.e. it is checked that *velocities*[i], converted to internal units, is in [0 .. `f_max_velocity_v`[*axes*[i]]].
 - in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller type: [-`f_max_velocity_v` .. `f_max_velocity_v`], i.e. it is checked that *velocities*[i], converted to internal units, is in [-`f_max_velocity_v`[*axes*[i]] .. `f_max_velocity_v`[*axes*[i]]].
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a `SDH::cSDHErrorInvalidParameter*` exception is thrown.

See also [SetAxisTargetVelocity\(int,double\)](#) for an overloaded variant to set a single axis target velocity or to set the same target velocity for all axes.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set target axis velocity of all axes to the given values:
std::vector<double> all_velocities;
all_velocities.push_back( 0.0 );
all_velocities.push_back( 11.0 );
all_velocities.push_back( 22.0 );
all_velocities.push_back( 33.0 );
all_velocities.push_back( 44.0 );
all_velocities.push_back( 55.0 );
all_velocities.push_back( 66.0 );

hand.SetAxisTargetVelocity( hand.all_axes, all_velocities );

// Set target axis velocity of axis 3 to 42 degrees per second:
hand.SetAxisTargetVelocity( 3, 42.0 );

// Set target velocity of for axis 0,4 and 2 to 0.0, 44.4 and 2.22 degr
ees per second respectively:
std::vector<int> axes042;
axes042.push_back( 0 );
axes042.push_back( 4 );
axes042.push_back( 2 );
std::vector<double> velocities042;
velocities042.push_back( 0.0 );
velocities042.push_back( 44.4 );
velocities042.push_back( 2.22 );

hand.SetAxisTargetVelocity( axes042, velocities042 );

// Set target axis velocity of all axes to 47.11 degrees per second
hand.SetAxisTargetVelocity( hand.All, 47.11 );
```

10.20.4.89 `std::vector< double > cSDH::SetAxisValueVector (std::vector< int > const & axes, std::vector< double > const & values, pSetFunction ll_set, pGetFunction ll_get, cUnitConverter const * uc, std::vector< double > const & min_values, std::vector< double > const & max_values, char const * name) throw (cSDHLibraryException*)` [protected]

Generic set function: set some given axes to given values

Parameters

<i>axes</i>	- a vector of axis indices
<i>values</i>	- a vector of values
<i>ll_set</i>	- a pointer to the low level set function to use
<i>ll_get</i>	- a pointer to the low level get function to use (for those axes where the given value is NaN)
<i>uc</i>	- a pointer to the unit converter object to use before sending values to <i>ll_set</i>

<i>min_values</i>	- a vector with the minimum allowed values in internal units
<i>max_values</i>	- a vector with the maximum allowed values in internal units
<i>name</i>	- a string with the name of the values (for constructing error message)

Returns

the values returned from the [SDH](#) will be returned (for most commands `ll_set/ll_get` functions this will be the *values*, except for the [cSDHSerial::tvav](#) and [cSDHSerial::tpap](#) functions)

Remarks

- The length of the *axis* and *values* vector must match.
- The indices can be given in any order, but the order of the elements of *axes* and *values* must match too. I.e. `values[i]` will be applied to axis `axes[i]` (not axis `i`)
- The indices are checked if they are valid axis indices.
- The values are checked if they are in the allowed range [*min_values* .. *f_max_values*], i.e. it is checked that `value[i]`, converted to the internal unit system by `uc->ToInternal()`, is in [*min_values*[*axes*[*i*]] .. *max_values*[*axes*[*i*]]].
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

Bug

Setting a single axis velocity only sets the velocities of all other axes to 0! So in order to address only some axes you must provide the desired velocity for all axes that you want to move in every call:

Bug

With SDH firmware 0.0.2.16 and SDHLibrary-CPP 0.0.2.3 and binary communication (see also [SDH_USE_BINARY_COMMUNICATION](#)) setting of single (more precisely: less-than-all) axis parameters (position, velocity, acceleration) does not work as expected: If a single parameter is to be set for a single axis then that parameter is set to 0 for all other axes.

Workaround: If you want to access several different axes one after another then you have to keep a vector of the parameter for all your used axes in your application. You can then update single values of the vector on demand, but you have to send the complete vector to the SDHlibrary functions on every call.

10.20.4.90 void cSDH::SetController (cSDHBase::eControllerType controller) throw (cSDHLibraryException*)

Set the type of axis controller to be used in the SDH

With SDH firmware $\geq 0.0.2.7$ this will automatically set valid default values for all target velocities, accelerations and positions in the SDH firmware, according to the *controller* type:

- **eCT_POSE:**
 - target velocities will be set to default (40 deg/s)
 - target accelerations will be set to default (100 deg/(s*s))
 - target positions will be set to default (0.0 deg)
- **eCT_VELOCITY:**
 - target velocities will be set to default (0 deg/s)
- **eCT_VELOCITY_ACCELERATION:**
 - target velocities will be set to default (0 deg/s)
 - target accelerations will be set to default (100 deg/(s*s))

This will also adjust the lower limits of the allowed velocities here in the SDHLibrary, since the eCT_POSE controller allows only positive velocities while the eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controllers require also negative velocities.

Attention

The availability of a controller type depends on the SDH firmware of the attached SDH and is checked here.

- firmware \leq 0.0.2.5: only eCT_POSE
- firmware \geq 0.0.2.6: eCT_POSE, eCT_VELOCITY, eCT_VELOCITY_ACCELERATION

Parameters

<i>controller</i>	- identifier of controller to set. Valid values are defined in eControllerType
-------------------	--

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Set the pose controller in the SDH
// (see e.g. demo-simple.cpp, demo-simple2.cpp, demo-simple3.cpp for further examples)
hand.SetController( hand.eCT_POSE );

// Set the simple velocity controller in the SDH:
hand.SetController( hand.eCT_VELOCITY );

// Set the velocity with acceleration ramp controller in the SDH:
// (see e.g. demo-velocity-acceleration.cpp for further examples)
hand.SetController( hand.eCT_VELOCITY_ACCELERATION );
```

10.20.4.91 `virtual void SDH::cSDH::SetDebugOutput (std::ostream * debuglog)`
`[inline, virtual]`

change the stream to use for debug messages

Reimplemented from [SDH::cSDHBase](#).

10.20.4.92 `void cSDH::SetFingerEnable (std::vector< int > const & fingers, std::vector< double > const & states) throw (cSDHLibraryException*)`

Set enabled/disabled state of axis controllers of finger(s).

The controllers of the axes of the selected fingers are enabled/disabled in the SDH. Disabled axes are not powered and thus might not remain in their current pose due to gravity, inertia or other external influences. But to prevent overheating the axis controllers should be switched of when not needed.

Parameters

<i>fingers</i>	- A vector of finger indices to access.
<i>states</i>	- A vector of enabled states (0 = disabled, !=0 = enabled) to set. If any of the numbers in the vector is NaN (Not a Number) then the currently set enabled state will be kept for the corresponding axis.

Remarks

- The lengths of the *fingers* and *states* vector must match.
- The indices can be given in any order, but the order of their elements must match, i.e. `state[i]` will be applied to finger `fingers[i]` (not finger `i`).
- The indices are checked if they are valid finger indices.
- If **any** index is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.
- As axis 0 is used for finger 0 and 2, axis 0 is disabled only if both finger 0 and 1 are disabled.

See also [SetFingerEnable\(int,double\)](#), [SetFingerEnable\(int,bool\)](#) for overloaded variants to set a single finger enabled/disabled or to set the same state for all fingers. See further [SetFingerEnable\(std::vector<int>const&,std::vector<bool>const&\)](#) for a variant that accepts a `bool` vector for the states to set.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Enable finger 1 and 2 while disabling finger 0 :
std::vector<double> states012;
states012.push_back( 0.0 );
states012.push_back( 1.0 );
states012.push_back( 1.0 );

hand.SetFingerEnable( hand.all_axes, states012 );
// (this will keep axis 0 (used by the disabled finger 0) enabled,
// since axis 0 is needed by the enabled finger 2 too);

// Enable all fingers:
hand.SetFingerEnable( hand.All,true );
```



```
// Disable all fingers:
hand.SetFingerEnable( hand.All, 0.0 );

// Disable finger 2:
hand.SetFingerEnable( 2, false );
```

10.20.4.93 void `cSDH::SetFingerEnable (std::vector< int > const & fingers, std::vector< bool > const & states) throw (cSDHLibraryException*)`

Like `SetFingerEnable(std::vector<int>const&,std::vector<double>const&)`, just with states as vector of `bool` values, see there for details and examples.

10.20.4.94 void `cSDH::SetFingerEnable (int iFinger, bool state) throw (cSDHLibraryException*)`

Like `SetFingerEnable(std::vector<int>const&,std::vector<double>const&)`, just for a single finger *iAxis* and a single angle *angle*, see there for details and examples.

10.20.4.95 void `cSDH::SetFingerEnable (int iFinger, double state = 1.0) throw (cSDHLibraryException*)`

Like `SetFingerEnable(std::vector<int>const&,std::vector<double>const&)`, just for a single finger *iAxis* and a single angle *angle*, see there for details and examples.

10.20.4.96 void `cSDH::SetFingerTargetAngle (int iFinger, std::vector< double > const & angles) throw (cSDHLibraryException*)`

Set the target angle(s) for a single finger.

The target axis angles *angle* of finger *iFinger* are stored in the SDH. The movement is not executed until an additional move command is sent.

Parameters

<i>iFinger</i>	- index of finger to access. This must be a single index.
<i>angles</i>	- the angle(s) to set or <code>None</code> to set the current actual axis angles of the finger as target angle. This can be a single number or a vector of numbers. The value(s) are expected in the configured angle unit system uc_angle .

Remarks

- Setting the target angles will **not** make the finger move.
- The *iFinger* index is checked if it is a valid finger index.
- The angles are checked if they are in the allowed range `[0 .. f_max_angle_v]`, i.e. it is checked that `angles[i]`, converted to internal units, is in `[0 .. f_max_angle_v[finger_axis_index[iFinger][i]]]`.
- If **any** index or value is invalid then **none** of the specified values is sent to the SDH, instead a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.

See also [SetFingerTargetAngle\(int,double,double,double\)](#) for an overloaded variant to set finger axis target angles from single double values.

Examples:

```
// Assuming "hand" is a cSDH object ...

// Set target axis angles of finger 0 to { 10.0, -08.15, 47.11 } degree
s
std::vector<double> angles;
angles.push_back( 10.0 );
angles.push_back( -08.15 );
angles.push_back( 47.11 );

hand.SetFingerTargetAngle( 0, angles );

// Set target axis angles of finger 1 to { 0.0, 24.7, 17.4 } degrees
angles[0] = 0.0; // "virtual" base axis of finger 1
angles[1] = 24.7;
angles[2] = 17.4;
hand.SetFingerTargetAngle( 1, {0.0, 24.7, 17.4 } );

// Set target axis angles of all axes of finger 0 to 12.34 degrees
hand.SetFingerTargetAngle( 0, 12.34, 12.34, 12.34 );

// REMARK: the last command changed the previously set target axis
// angle for axis 0, since axis 0 is used as base axis for both
// finger 0 and 2!
```

10.20.4.97 void **cSDH::SetFingerTargetAngle** (int *iFinger*, double *a0*, double *a1*, double *a2*)
throw (cSDHLibraryException*)

Like [SetFingerTargetAngle\(int,std::vector<double>const&\)](#), just with individual finger axis angles *a0*, *a1* and *a2*.

10.20.4.98 void **cSDH::SetVelocityProfile** (eVelocityProfile *velocity_profile*) throw
(cSDHLibraryException*)

Set the type of velocity profile to be used in the SDH

Parameters

<i>velocity_profile</i>	- Name or number of velocity profile to set. Valid values are defined in eVelocityProfileType
-------------------------	---

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Set the sin square velocity profile in the SDH:
hand.SetVelocityProfile( hand.eVP_SIN_SQUARE );
```

```
// Or else set the ramp velocity profile in the SDH:
hand.SetVelocityProfile( hand.eVP_RAMP )
```

10.20.4.99 void cSDH::Stop (void) throw (cSDHLibraryException*)

Stop movement of all axes but keep controllers on

This command will always be executed sequentially: it will return only after the SDH has confirmed the stop

Bug

For now this will **NOT** work while a [GripHand\(\)](#) command is executing, even if that was initiated non-sequentially!

Bug

With SDH firmware < 0.0.2.7 this made the axis jerk in eCT_POSE controller type. This is resolved in SDH firmware 0.0.2.7 for the eCT_POSE controller type with velocity profile eVP_RAMP. For the eCT_POSE controller type with velocity profile eVP_SIN_SQUARE changing target points/ velocities while moving will still make the axes jerk.

=> **Partly resolved in SDH firmware 0.0.2.7**

Examples:

```
// Assuming 'hand' is a cSDH object ...

// Perform a stop:
hand.Stop();
```

10.20.4.100 std::vector< int > cSDH::ToIndexVector (int index, std::vector< int > & all_replacement, int maxindex, char const * name) throw (cSDHLibraryException*) [protected]

Internal helper function: return a vector of checked indices according to index.

Parameters

<i>index</i>	- The index to vectorize or All
<i>all_replacement</i>	- a vector to return if <i>index</i> is All
<i>maxindex</i>	- the <i>index</i> is checked if in [0..\ maxindex[(i.e. not including <i>maxindex</i>)
<i>name</i>	- A name for the things index, used to report out of bounds errors

Returns

- If *index* is [All](#) then *all_replacement* is returned.
- If *index* is a single number >= 0 then it is checked if in [0..\ maxindex[and

a vector of length 1 is returned containing only *index*.

- In case *index* exceeds *maxindex* a (cSDHErrorInvalidParameter*) exception is thrown.

10.20.4.101 void cSDH::UseDegrees (void)

Shortcut to set the unit system to degrees.

After calling this (axis) angles are set/reported in degrees and angular velocities are set/reported in degrees/second

Examples:

```
// Assuming 'hand' is a cSDH object ...

// make hand object use degrees and degrees/second for angles and angular
// velocities
hand.UseDegrees();
// as degrees, degrees/second are the default this is needed only if the
// unit system was changed before
```

10.20.4.102 void cSDH::UseRadians (void)

Shortcut to set the unit system to radians.

After calling this axis angles are set/reported in radians and angular velocities are set/reported in radians/second

Examples:

```
// Assuming 'hand' is a cSDH object ...

// make hand object use radians and radians/second for angles and angular
// velocities
hand.UseRadians();
```

10.20.4.103 void cSDH::WaitAxis (int iAxis, double timeout = -1.0) throw (cSDHLibraryException*)

Like [WaitAxis\(std::vector<int>const&,double\)](#), just for a single axis *iAxis*, see there for details and examples.

If *iAxis* is [All](#) then wait for all axes.

10.20.4.104 void cSDH::WaitAxis (std::vector< int > const & axes, double timeout = -1.0) throw (cSDHLibraryException*)

Wait until the movement(s) of axis(axes) has finished

The state of the given axis(axes) is(are) queried until all axes are no longer moving.

Parameters

<i>axes</i>	- A vector of axis indices to access.
<i>timeout</i>	- a timeout in seconds or -1.0 (default) to wait indefinitely.

- The indices in *axes* are checked if they are valid axis indices.
- If **any** axis index is invalid then a [SDH::cSDHErrorInvalidParameter*](#) exception is thrown.
 - If *timeout* < 0 then this function will wait arbitrarily long
 - If a *timeout* is given then this function will throw a [cSDHErrorCommunication](#) exception if the given axes are still moving after *timeout* many seconds

See also [WaitAxis\(int,double\)](#) for an overloaded variant to wait for a single axis or all axes.

Bug

Due to a bug in SDH firmwares prior to 0.0.2.6 the [WaitAxis\(\)](#) command was somewhat unreliable there. When called immediately after a movement command like [MoveHand\(\)](#), then the [WaitAxis\(\)](#) command returned immediately without waiting for the end of the movement. With SDH firmwares 0.0.2.6 and newer this is no longer problematic and [WaitAxis\(\)](#) works as expected.

=> **Resolved in SDH firmware 0.0.2.6**

Bug

With SDH firmware 0.0.2.6 [WaitAxis\(\)](#) did not work if one of the new velocity based controllers (eCT_VELOCITY, eCT_VELOCITY_ACCELERATION) was used. With SDH firmwares 0.0.2.7 and newer this now works. Here the [WaitAxis\(\)](#) waits until the selected axes come to velocity 0.0

=> **Resolved in SDH firmware 0.0.2.7**

Examples:

Example 1, WaitAxis and eCT_POSE controller, see also the demo program demo-simple3:

```
// Assuming "hand" is a cSDH object ...

hand.SetController( eCT_POSE );

// Set a new target pose for axis 1,2 and 3
std::vector<int> axes123;
axes123.push_back( 1 );
axes123.push_back( 2 );
axes123.push_back( 3 );

std::vector<double> angles123;
angles123.push_back( -20.0 );
angles123.push_back( -30.0 );
angles123.push_back( -40.0 );
```

```

hand.SetAxisTargetAngle( axes123, angles123 );

// Move axes there non sequentially:
hand.MoveAxis( axes123, false );

// The last call returned immediately so we now have time to
// do something else while the hand is moving:

// ... insert any calculation here ...

// Before doing something else with the hand make sure the
// selected axes have finished the last movement:
hand.WaitAxis( axes123 );

// go back home (all angles to 0.0):
hand.SetAxisTargetAngle( hand.All, 0.0 );

// Move all axes there non sequentially:
hand.MoveAxis( hand.All, False );

// ... insert any other calculation here ...

// Wait until all axes are there, with a timeout of 10s:
hand.WaitAxis( hand.All, 10.0 );

// now we are at the desired position.

```

Example 2, WaitAxis and eCT_VELOCITY_ACCELERATION controller, see also the demo program demo-velocity-acceleration

```

// Assuming "hand" is a cSDH object ...

hand.SetController( eCT_VELOCITY_ACCELERATION );

// Set a new target velocity for axis 1,2 and 3
std::vector<int> axes123;
axes123.push_back( 1 );
axes123.push_back( 2 );
axes123.push_back( 3 );

std::vector<double> velocities123;
velocities123.push_back( -20.0 );
velocities123.push_back( -30.0 );
velocities123.push_back( -40.0 );

hand.SetAxisTargetVelocity( axes123, velocities123 ); // this will make
the axes move!

// The last call returned immediately so we now have time to
// do something else while the hand is moving:

// ... insert any calculation here ...

// to break and stop the movement just set the target velocities to 0.0

velocities123[0] = 0.0;
velocities123[1] = 0.0;
velocities123[2] = 0.0;

```

```

    hand.SetAxisTargetVelocity( axes123, velocities123 ); // this will make
    the axes break with the default (de)acceleration

    // The previous command returned immediately, so
    // before doing something else with the hand make sure the
    // selected axes have stopped:
    hand.WaitAxis( axes123 );

    // now the axes have stopped

```

10.20.5 Member Data Documentation

10.20.5.1 `std::vector<int> SDH::cSDH::all_axes`

A vector with indices of all axes (in natural order), including the virtual axis.

10.20.5.2 `std::vector<int> SDH::cSDH::all_fingers`

A vector with indices of all fingers (in natural order)

10.20.5.3 `std::vector<int> SDH::cSDH::all_real_axes`

A vector with indices of all real axes (in natural order), excluding the virtual axis.

10.20.5.4 `std::vector<int> SDH::cSDH::all_temperature_sensors`

A vector with indices of all temperature sensors.

10.20.5.5 `cSerialBase* SDH::cSDH::com` [protected]

10.20.5.6 `cSDHSerial SDH::cSDH::comm_interface`

The object to interface with the SDH attached via serial RS232 or CAN or TCP.

10.20.5.7 `double SDH::cSDH::d` [protected]

10.20.5.8 `std::vector<double> SDH::cSDH::f_max_acceleration_v` [protected]

Maximum allowed axis acceleration (in internal units (degrees/(second * second))), including the virtual axis.

10.20.5.9 `std::vector<double> SDH::cSDH::f_max_angle_v` [protected]

Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.

10.20.5.10 `std::vector<double> SDH::cSDH::f_max_motor_current_v`
[protected]

Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.

10.20.5.11 `std::vector<double> SDH::cSDH::f_max_velocity_v` [protected]

Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

10.20.5.12 `std::vector<double> SDH::cSDH::f_min_acceleration_v`
[protected]

Minimum allowed axis acceleration (in internal units (degrees/(second * second))), including the virtual axis.

10.20.5.13 `std::vector<double> SDH::cSDH::f_min_angle_v` [protected]

Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.

10.20.5.14 `std::vector<double> SDH::cSDH::f_min_motor_current_v`
[protected]

Minimum allowed motor currents (in internal units (Ampere)), including the virtual axis.

10.20.5.15 `std::vector<double> SDH::cSDH::f_min_velocity_v` [protected]

Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

10.20.5.16 `std::vector<double> SDH::cSDH::f_ones_v` [protected]

Vector of 3 1.0 values.

10.20.5.17 `std::vector<double> SDH::cSDH::f_zeros_v` [protected]

Vector of 3 epsilon values.

Vector of 3 0.0 values

10.20.5.18 `std::vector<std::vector<int> > SDH::cSDH::finger_axis_index`
[protected]

Mapping of finger index, finger axis index to axis index:

10.20.5.19 `std::vector<int> SDH::cSDH::finger_number_of_axes` [protected]

Mapping of finger index to number of real axes of fingers:

10.20.5.20 `double SDH::cSDH::grip_max_velocity` [protected]

Maximum allowed grip velocity (in internal units (degrees/second))

10.20.5.21 `double SDH::cSDH::h` [protected]

10.20.5.22 `double SDH::cSDH::l1` [protected]

length of limb 1 (proximal joint to distal joint) in mm

10.20.5.23 `double SDH::cSDH::l2` [protected]

length of limb 2 (distal joint to fingertip) in mm

10.20.5.24 `int SDH::cSDH::nb_all_axes` [protected]

The number of all axes including virtual axes.

10.20.5.25 `int SDH::cSDH::NUMBER_OF_AXES_PER_FINGER`
[protected]

The number of axis per finger (for finger 1 this includes the "virtual" base axis)

10.20.5.26 `int SDH::cSDH::NUMBER_OF_VIRTUAL_AXES` [protected]

The number of virtual axes.

10.20.5.27 `std::vector<std::vector<double> > SDH::cSDH::offset` [protected]

list of xyz-vectors for all fingers with offset from (0,0,0) of proximal joint in mm

10.20.5.28 `std::vector<double> SDH::cSDH::ones_v` [protected]

Vector of nb_all_axes 1.0 values.

10.20.5.29 const cUnitConverter* SDH::cSDH::uc_angle

unit convert for (axis) angles: default = [SDH::cSDH::uc_angle_degrees](#)

10.20.5.30 cUnitConverter const cSDH::uc_angle_degrees [static]

Default converter for angles (internal unit == external unit): degrees.

10.20.5.31 cUnitConverter const cSDH::uc_angle_radians [static]

Converter for angles: external unit = radians.

10.20.5.32 const cUnitConverter* SDH::cSDH::uc_angular_acceleration

unit convert for (axis) angular accelerations: default = [SDH::cSDH::uc_angular_acceleration_degrees_per_second_squared](#)

10.20.5.33 cUnitConverter const cSDH::uc_angular_acceleration_degrees_per_second_squared [static]

Default converter for angular accelerations (internal unit == external unit): degrees / second.

10.20.5.34 cUnitConverter const cSDH::uc_angular_acceleration_radians_per_second_squared [static]

Converter for angular velocities: external unit = radians/second.

10.20.5.35 const cUnitConverter* SDH::cSDH::uc_angular_velocity

unit convert for (axis) angular velocities: default = [SDH::cSDH::uc_angular_velocity_degrees_per_second](#)

10.20.5.36 cUnitConverter const cSDH::uc_angular_velocity_degrees_per_second [static]

Default converter for angular velocities (internal unit == external unit): degrees / second.

10.20.5.37 cUnitConverter const cSDH::uc_angular_velocity_radians_per_second [static]

Converter for angular velocities: external unit = radians/second.

10.20.5.38 const cUnitConverter* SDH::cSDH::uc_motor_current

unit converter for motor current: default = [SDH::cSDH::uc_motor_current_ampere](#)

10.20.5.39 cUnitConverter const cSDH::uc_motor_current_ampere [static]

Default converter for motor current (internal unit == external unit): Ampere.

10.20.5.40 cUnitConverter const cSDH::uc_motor_current_milliampere [static]

Converter for motor current: external unit = milli Ampere.

10.20.5.41 const cUnitConverter* SDH::cSDH::uc_position

unit converter for position: default = [SDH::cSDH::uc_position_millimeter](#)

10.20.5.42 cUnitConverter const cSDH::uc_position_meter [static]

Converter for position: external unit = meter.

10.20.5.43 cUnitConverter const cSDH::uc_position_millimeter [static]

Default converter for position (internal unit == external unit): millimeter.

10.20.5.44 const cUnitConverter* SDH::cSDH::uc_temperature

unit convert for temperatures: default = [SDH::cSDH::uc_temperature_celsius](#)

10.20.5.45 cUnitConverter const cSDH::uc_temperature_celsius [static]

Default converter for temperatures (internal unit == external unit): degrees celsius.

10.20.5.46 cUnitConverter const cSDH::uc_temperature_fahrenheit [static]

Converter for temperatures: external unit = degrees fahrenheit.

10.20.5.47 const cUnitConverter* SDH::cSDH::uc_time

unit convert for times: default = uc_time_seconds

10.20.5.48 `cUnitConverter` `const cSDH::uc_time_milliseconds` `[static]`

Converter for times: external unit = milliseconds.

10.20.5.49 `cUnitConverter` `const cSDH::uc_time_seconds` `[static]`

Default converter for times (internal unit == external unit): seconds.

10.20.5.50 `std::vector<double>` `SDH::cSDH::zeros_v` `[protected]`

Vector of `nb_all_axes` 0.0 values.

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

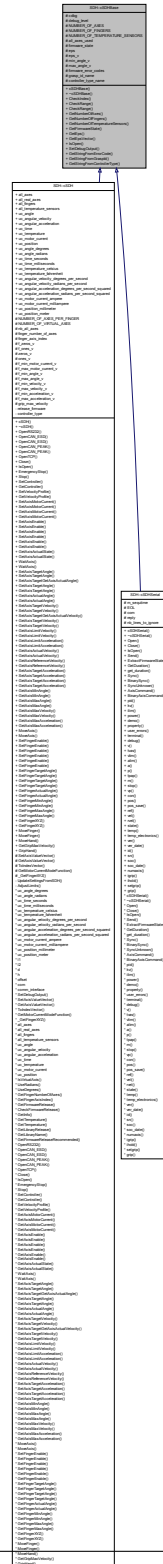
- [sdh/sdh.h](#)
- [sdh/sdh.cpp](#)

10.21 `SDH::cSDHBase` Class Reference

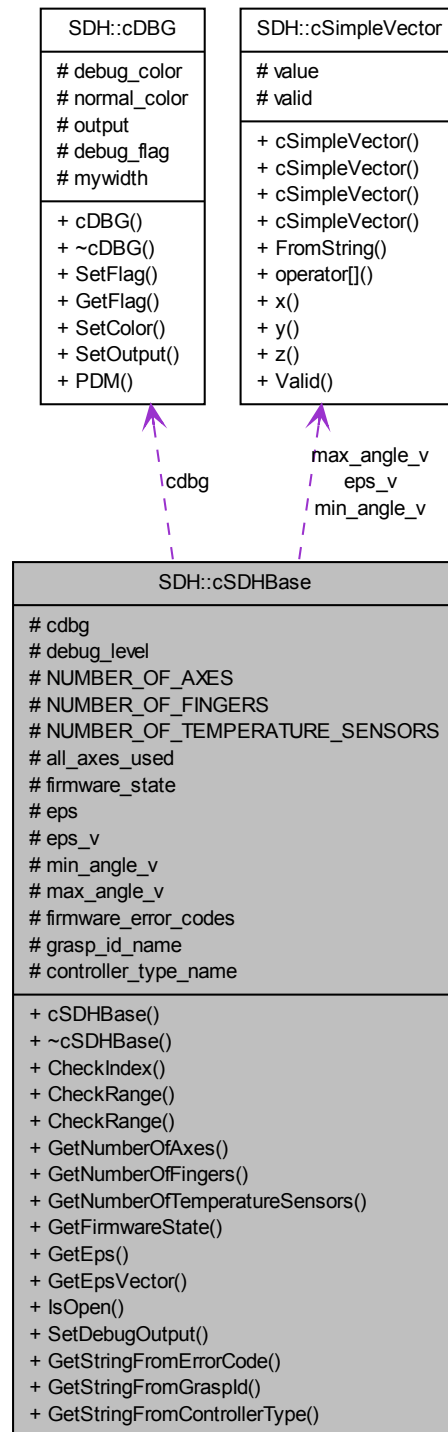
The base class to control the SCHUNK Dexterous Hand.

```
#include <sdhbase.h>
```

Inheritance diagram for SDH::cSDHBase:



Collaboration diagram for SDH::cSDHBase:



Public Types

- enum { [All](#) = -1 }
Anonymous enum (instead of define like macros)
- enum [eErrorCode](#) {
[eEC_SUCCESS](#) = 0, [eEC_NOT_AVAILABLE](#), [eEC_NOT_INITIALIZED](#), [eEC_ALREADY_RUNNING](#),
[eEC_FEATURE_NOT_SUPPORTED](#), [eEC_INCONSISTENT_DATA](#), [eEC_TIMEOUT](#),
[eEC_READ_ERROR](#),
[eEC_WRITE_ERROR](#), [eEC_INSUFFICIENT_RESOURCES](#), [eEC_CHECKSUM_ERROR](#), [eEC_NOT_ENOUGH_PARAMS](#),
[eEC_NO_PARAMS_EXPECTED](#), [eEC_CMD_UNKNOWN](#), [eEC_CMD_FORMAT_ERROR](#), [eEC_ACCESS_DENIED](#),
[eEC_ALREADY_OPEN](#), [eEC_CMD_FAILED](#), [eEC_CMD_ABORTED](#), [eEC_INVALID_HANDLE](#),
[eEC_DEVICE_NOT_FOUND](#), [eEC_DEVICE_NOT_OPENED](#), [eEC_IO_ERROR](#),
[eEC_INVALID_PARAMETER](#),
[eEC_RANGE_ERROR](#), [eEC_NO_DATAPIPE](#), [eEC_INDEX_OUT_OF_BOUNDS](#),
[eEC_HOMING_ERROR](#),
[eEC_AXIS_DISABLED](#), [eEC_OVER_TEMPERATURE](#), [eEC_MAX_COMMANDS_EXCEEDED](#), [eEC_INVALID_PASSWORD](#),
[eEC_MAX_COMMANDLINE_EXCEEDED](#), [eEC_CRC_ERROR](#), [eEC_NO_COMMAND](#), [eEC_INTERNAL](#),
[eEC_UNKNOWN_ERROR](#), [eEC_DIMENSION](#) }
- enum [eGraspId](#) {
[eGID_INVALID](#) = -1, [eGID_CENTRICAL](#) = 0, [eGID_PARALLEL](#) = 1, [eGID_CYLINDRICAL](#) = 2,
[eGID_SPHERICAL](#) = 3, [eGID_DIMENSION](#) }
The enum values of the known grasps.
- enum [eControllerType](#) {
[eCT_INVALID](#) = -1, [eCT_POSE](#) = 0, [eCT_VELOCITY](#), [eCT_VELOCITY_ACCELERATION](#),
[eCT_DIMENSION](#) }
An enum for all possible SDH internal controller types (order must match that in the SDH firmware in inc/sdh2.h)
- enum [eVelocityProfile](#) { [eVP_INVALID](#) = -1, [eVP_SIN_SQUARE](#), [eVP_RAMP](#),
[eVP_DIMENSION](#) }
An enum for all possible SDH internal velocity profile types.

Public Member Functions

- [cSDHBase](#) (int [debug_level](#))
- virtual [~cSDHBase](#) ()
- void [CheckIndex](#) (int index, int maxindex, char const *name="") throw (cSDHErrorInvalidParameter*)

Check if index is in [0 .. maxindex-1] or All. Throw a [cSDHErrorInvalidParameter](#) exception if not.

- void [CheckRange](#) (double value, double minvalue, double maxvalue, char const *name="") throw (cSDHErrorInvalidParameter*)

Check if value is in [minvalue .. maxvalue]. Throw a [cSDHErrorInvalidParameter](#) exception if not.

- void [CheckRange](#) (double *values, double *minvalues, double *maxvalues, char const *name="") throw (cSDHErrorInvalidParameter*)

Check if any value[i] in array values is in [minvalue[i] .. maxvalue[i]]. Throw a [cSDHErrorInvalidParameter](#) exception if not.

- int [GetNumberOfAxes](#) (void)

Return the number of axes of the SDH.

- int [GetNumberOfFingers](#) (void)

Return the number of fingers of the SDH.

- int [GetNumberOfTemperatureSensors](#) (void)

Return the number of temperature sensors of the SDH.

- [eErrorCode](#) [GetFirmwareState](#) (void)

Return the last known state of the SDH firmware.

- double [GetEps](#) (void)

Return the [eps](#) value.

- [cSimpleVector](#) const & [GetEpsVector](#) (void)

Return simple vector of number of axes epsilon values.

- virtual bool [IsOpen](#) (void)=0

Return true if connection to SDH firmware/hardware is open.

- virtual void [SetDebugOutput](#) (std::ostream *debuglog)

change the stream to use for debug messages

Static Public Member Functions

- static char const * [GetStringFromErrorCode](#) (eErrorCode error_code)
Return a ptr to a (static) string describing error code error_code.
- static char const * [GetStringFromGraspId](#) (eGraspId grasp_id)
Return a ptr to a (static) string describing grasp id grasp_id.
- static char const * [GetStringFromControllerType](#) (eControllerType controller_type)
Return a ptr to a (static) string describing controller type controller_Type.

Protected Attributes

- [cDBG cdbg](#)
debug stream to print colored debug messages
- int [debug_level](#)
debug level of this object
- int [NUMBER_OF_AXES](#)
The number of axes.
- int [NUMBER_OF_FINGERS](#)
The number of fingers.
- int [NUMBER_OF_TEMPERATURE_SENSORS](#)
The number of temperature sensors.
- int [all_axes_used](#)
Bit field with the bits for all axes set.
- eErrorCode [firmware_state](#)
the last known state of the SDH firmware
- double [eps](#)
epsilon value (max absolute deviation of reported values from actual hardware values) (needed since SDH firmware limits number of digits reported)
- cSimpleVector [eps_v](#)
simple vector of 7 epsilon values
- cSimpleVector [min_angle_v](#)
simple vector of 7 0 values ???

- [cSimpleVector max_angle_v](#)

Maximum allowed axis angles (in internal units (degrees))

Static Protected Attributes

- static char const * [firmware_error_codes](#) []

A mapping from [eErrorCode](#) error code enums to strings with human readable error messages.

- static char const * [grasp_id_name](#) []

A mapping from [eGraspId](#) grasp id enums to strings with human readable grasp id names.

- static char const * [controller_type_name](#) []

A mapping from [eControllerType](#) controller type enums to strings with human readable controller type names.

10.21.1 Detailed Description

The base class to control the SCHUNK Dexterous Hand. End-Users should **NOT** use this class directly, as it only provides some common settings and no function interface. End users should use the class [cSDH](#) instead, as it provides the end-user functions to control the SDH.

10.21.2 Member Enumeration Documentation

10.21.2.1 anonymous enum

Anonymous enum (instead of define like macros)

Enumerator:

All A meta-value that means "access all possible values".

10.21.2.2 enum [SDH::cSDHBase::eControllerType](#)

An enum for all possible SDH internal controller types (order must match that in the SDH firmware in inc/sdh2.h)

Enumerator:

eCT_INVALID invalid controller_type (needed for [cSDHSerial::con\(\)](#) to indicate "read current controller type")

eCT_POSE coordinated position controller (position per axis => "pose controller"), all axes start and stop moving at the same time

eCT_VELOCITY velocity controller, velocities of axes are controlled independently (not implemented in SDH firmwares up to and including 0.0.2.5)

eCT_VELOCITY_ACCELERATION velocity controller with acceleration ramp, velocities and accelerations of axes are controlled independently (not implemented in SDH firmwares up to and including 0.0.2.5)

eCT_DIMENSION Endmarker and dimension.

10.21.2.3 enum SDH::cSDHBase::eErrorCode

The error codes of the SDH firmware

Enumerator:

eEC_SUCCESS
eEC_NOT_AVAILABLE
eEC_NOT_INITIALIZED
eEC_ALREADY_RUNNING
eEC_FEATURE_NOT_SUPPORTED
eEC_INCONSISTENT_DATA
eEC_TIMEOUT
eEC_READ_ERROR
eEC_WRITE_ERROR
eEC_INSUFFICIENT_RESOURCES
eEC_CHECKSUM_ERROR
eEC_NOT_ENOUGH_PARAMS
eEC_NO_PARAMS_EXPECTED
eEC_CMD_UNKNOWN
eEC_CMD_FORMAT_ERROR
eEC_ACCESS_DENIED
eEC_ALREADY_OPEN
eEC_CMD_FAILED
eEC_CMD_ABORTED
eEC_INVALID_HANDLE
eEC_DEVICE_NOT_FOUND
eEC_DEVICE_NOT_OPENED
eEC_IO_ERROR
eEC_INVALID_PARAMETER
eEC_RANGE_ERROR

eEC_NO_DATAPIPE
eEC_INDEX_OUT_OF_BOUNDS
eEC_HOMING_ERROR
eEC_AXIS_DISABLED
eEC_OVER_TEMPERATURE
eEC_MAX_COMMANDS_EXCEEDED
eEC_INVALID_PASSWORD
eEC_MAX_COMMANDLINE_EXCEEDED
eEC_CRC_ERROR
eEC_NO_COMMAND
eEC_INTERNAL
eEC_UNKNOWN_ERROR
eEC_DIMENSION Endmarker and dimension.

10.21.2.4 enum SDH::cSDHBase::eGraspId

The enum values of the known grasps.

Enumerator:

eGID_INVALID invalid grasp id
eGID_CENTRICAL central grasp: ???
eGID_PARALLEL parallel grasp: ???
eGID_CYLINDRICAL cylindrical grasp: ???
eGID_SPHERICAL spherecial grasp: ???
eGID_DIMENSION Endmarker and dimension.

10.21.2.5 enum SDH::cSDHBase::eVelocityProfile

An enum for all possible SDH internal velocity profile types.

Enumerator:

eVP_INVALID not a valid velocity profile, used to make [SDH::cSDHSerial::vp\(\)](#) read the velocity profile from the SDH firmware
eVP_SIN_SQUARE sin square velocity profile
eVP_RAMP ramp velocity profile
eVP_DIMENSION endmarker and dimension

10.21.3 Constructor & Destructor Documentation

10.21.3.1 cSDHBase::cSDHBase (int *debug_level*)

Constructor of [cSDHBase](#) class, initialize internal variables and settings

Parameters

<i>debug_level</i>	: debug level of the created object. If the <i>debug_level</i> of an object is > 0 then it will output debug messages. <ul style="list-style-type: none"> (Subclasses of cSDHBase like cSDH or cSDHSerial use additional settings, see there.)
--------------------	--

10.21.3.2 virtual SDH::cSDHBase::~~cSDHBase () [inline, virtual]

virtual destructor to make compiler happy

10.21.4 Member Function Documentation

10.21.4.1 void cSDHBase::CheckIndex (int *index*, int *maxindex*, char const * *name* = " ") throw (cSDHErrorInvalidParameter*)

Check if *index* is in $[0 .. \text{maxindex}-1]$ or All. Throw a [cSDHErrorInvalidParameter](#) exception if not.

10.21.4.2 void cSDHBase::CheckRange (double *value*, double *minvalue*, double *maxvalue*, char const * *name* = " ") throw (cSDHErrorInvalidParameter*)

Check if *value* is in $[\text{minvalue} .. \text{maxvalue}]$. Throw a [cSDHErrorInvalidParameter](#) exception if not.

10.21.4.3 void cSDHBase::CheckRange (double * *values*, double * *minvalues*, double * *maxvalues*, char const * *name* = " ") throw (cSDHErrorInvalidParameter*)

Check if any value[i] in array *values* is in $[\text{minvalue}[i] .. \text{maxvalue}[i]]$. Throw a [cSDHErrorInvalidParameter](#) exception if not.

10.21.4.4 double cSDHBase::GetEps (void)

Return the [eps](#) value.

10.21.4.5 cSimpleVector const & cSDHBase::GetEpsVector (void)

Return simple vector of number of axes epsilon values.

10.21.4.6 cSDHBase::eErrorCode cSDHBase::GetFirmwareState (void)

Return the last known state of the SDH firmware.

10.21.4.7 int cSDHBase::GetNumberOfAxes (void)

Return the number of axes of the SDH.

10.21.4.8 int cSDHBase::GetNumberOfFingers (void)

Return the number of fingers of the SDH.

10.21.4.9 int cSDHBase::GetNumberOfTemperatureSensors (void)

Return the number of temperature sensors of the SDH.

10.21.4.10 char const * cSDHBase::GetStringFromControllerType (eControllerType controller_type) [static]

Return a ptr to a (static) string describing controller type *controller_Type*.

10.21.4.11 char const * cSDHBase::GetStringFromErrorCode (eErrorCode error_code) [static]

Return a ptr to a (static) string describing error code *error_code*.

10.21.4.12 char const * cSDHBase::GetStringFromGraspId (eGraspId grasp_id) [static]

Return a ptr to a (static) string describing grasp id *grasp_id*.

10.21.4.13 virtual bool SDH::cSDHBase::IsOpen (void) [pure virtual]

Return true if connection to SDH firmware/hardware is open.

Implemented in [SDH::cSDH](#), and [SDH::cSDHSerial](#).

10.21.4.14 virtual void SDH::cSDHBase::SetDebugOutput (std::ostream * debuglog) [inline, virtual]

change the stream to use for debug messages

Reimplemented in [SDH::cSDH](#).

10.21.5 Member Data Documentation

10.21.5.1 `int SDH::cSDHBase::all_axes_used` [protected]

Bit field with the bits for all axes set.

10.21.5.2 `cDBG SDH::cSDHBase::cdbg` [protected]

debug stream to print colored debug messages

10.21.5.3 `char const * cSDHBase::controller_type_name` [static, protected]

Initial value:

```
{
    "eCT_POSE: position/pose controller coordinated",
    "eCT_VELOCITY: velocity controller",
    "eCT_VELOCITY_ACCELERATION: velocity with acceleration ramp controller",

    "eCT_DIMENSION: number of controller types"
}
```

A mapping from [eControllerType](#) controller type enums to strings with human readable controller type names.

10.21.5.4 `int SDH::cSDHBase::debug_level` [protected]

debug level of this object

10.21.5.5 `double SDH::cSDHBase::eps` [protected]

epsilon value (max absolute deviation of reported values from actual hardware values) (needed since SDH firmware limits number of digits reported)

10.21.5.6 `cSimpleVector SDH::cSDHBase::eps_v` [protected]

simple vector of 7 epsilon values

10.21.5.7 `char const * cSDHBase::firmware_error_codes` [static, protected]

Initial value:

```

{
    "eEC_SUCCESS: No error",
    "eEC_NOT_AVAILABLE: Service or data is not available",
    "eEC_NOT_INITIALIZED: The device is not initialized",
    "eEC_ALREADY_RUNNING: Data acquisition: the acquisition loop is already running",
    "eEC_FEATURE_NOT_SUPPORTED: The asked feature is not supported",
    "eEC_INCONSISTENT_DATA: One or more dependent parameters mismatch",
    "eEC_TIMEOUT: Timeout error",
    "eEC_READ_ERROR: Error while reading from a device",
    "eEC_WRITE_ERROR: Error while writing to a device",
    "eEC_INSUFFICIENT_RESOURCES: No memory available",
    "eEC_CHECKSUM_ERROR: Checksum error",
    "eEC_NOT_ENOUGH_PARAMS: Not enough parameters",
    "eEC_NO_PARAMS_EXPECTED: No parameters expected",
    "eEC_CMD_UNKNOWN: Unknown command",
    "eEC_CMD_FORMAT_ERROR: Command format error",
    "eEC_ACCESS_DENIED: Access denied",
    "eEC_ALREADY_OPEN: Interface already open",
    "eEC_CMD_FAILED: Command failed",
    "eEC_CMD_ABORTED: Command aborted",
    "eEC_INVALID_HANDLE: Invalid handle",
    "eEC_DEVICE_NOT_FOUND: Device not found",
    "eEC_DEVICE_NOT_OPENED: Device not open",
    "eEC_IO_ERROR: General I/O-Error",
    "eEC_INVALID_PARAMETER: Invalid parameter",
    "eEC_RANGE_ERROR: Range error",
    "eEC_NO_DATAPIPE: No datapipe was found to open the specified device path",
    "eEC_INDEX_OUT_OF_BOUNDS: The passed index is out of bounds",
    "eEC_HOMING_ERROR: Error while homing",
    "eEC_AXIS_DISABLED: The selected axis is disabled",
    "eEC_OVER_TEMPERATURE: Over-temperature",
    "eEC_MAX_COMMANDS_EXCEEDED: E_MAX_COMMANDS_EXCEEDED: cannot add more than CI_MAX_COMMANDS to interpreter / POSCON_MAX_OSCILLOSCOPE parameters to oscilloscope",
    "eEC_INVALID_PASSWORD: E_INVALID_PASSWORD: invalid password given for change user command",
    "eEC_MAX_COMMANDLINE_EXCEEDED: E_MAX_COMMANDLINE_EXCEEDED: the command line given is too long",
    "eEC_CRC_ERROR: Cyclic Redundancy Code error",
    "eEC_NO_COMMAND: No command available",

    "eEC_INTERNAL: internal error",
    "eEC_UNKNOWN_ERROR: unknown error",

    "eEC_DIMENSION: Number of error codes"
}

```

A mapping from [eErrorCode](#) error code enums to strings with human readable error messages.

10.21.5.8 eErrorCode SDH::cSDHBase::firmware_state [protected]

the last known state of the SDH firmware

10.21.5.9 `char const * cSDHBase::grasp_id_name` [static, protected]

Initial value:

```
{
    "eGID_CENTRICAL: central grasp",
    "eGID_PARALLEL: parallel grasp",
    "eGID_CYLINDRICAL: cylindrical grasp",
    "eGID_SPHERICAL: spherecial grasp",

    "eGID_DIMENSION: number of predefined grasp ids"
}
```

A mapping from [eGraspId](#) grasp id enums to strings with human readable grasp id names.

10.21.5.10 `cSimpleVector SDH::cSDHBase::max_angle_v` [protected]

Maximum allowed axis angles (in internal units (degrees))

10.21.5.11 `cSimpleVector SDH::cSDHBase::min_angle_v` [protected]

simple vector of 7 0 values ???

simple vector of 7 1 values ??? Minimum allowed axis angles (in internal units (degrees))

10.21.5.12 `int SDH::cSDHBase::NUMBER_OF_AXES` [protected]

The number of axes.

10.21.5.13 `int SDH::cSDHBase::NUMBER_OF_FINGERS` [protected]

The number of fingers.

10.21.5.14 `int SDH::cSDHBase::NUMBER_OF_TEMPERATURE_SENSORS`
[protected]

The number of temperature sensors.

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

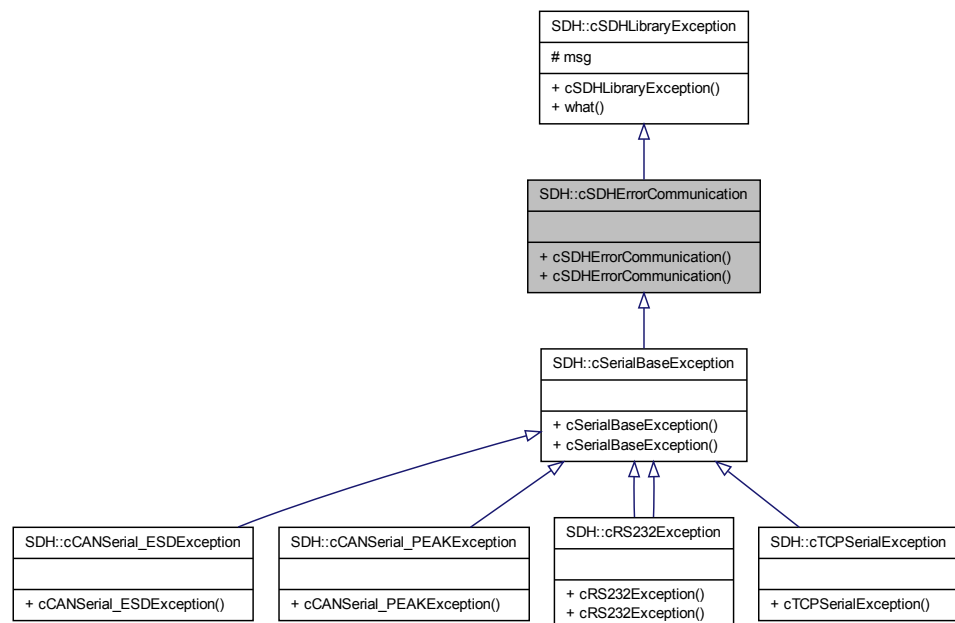
- [sdh/sdhbase.h](#)
- [sdh/sdhbase.cpp](#)

10.22 SDH::cSDHErrorCommunication Class Reference

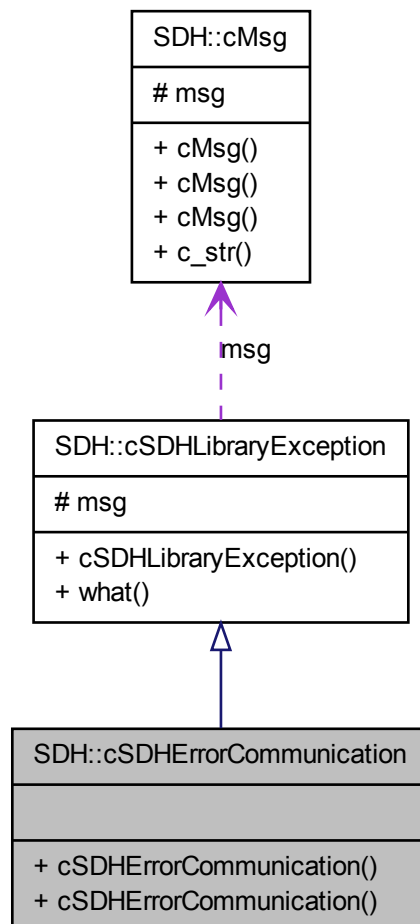
Derived exception class for exceptions related to communication between the SDHLi-
brary and the SDH.

```
#include <sdhexception.h>
```

Inheritance diagram for SDH::cSDHErrorCommunication:



Collaboration diagram for SDH::cSDHErrorCommunication:



Public Member Functions

- [cSDHErrorCommunication](#) ([cMsg](#) const &_msg)
- [cSDHErrorCommunication](#) (char const *_type, [cMsg](#) const &_msg)

10.22.1 Detailed Description

Derived exception class for exceptions related to communication between the SDHLibrary and the SDH.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 `SDH::cSDHErrorCommunication::cSDHErrorCommunication (cMsg const & _msg)`
[inline]

10.22.2.2 `SDH::cSDHErrorCommunication::cSDHErrorCommunication (char const * _type, cMsg const & _msg)` [inline]

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

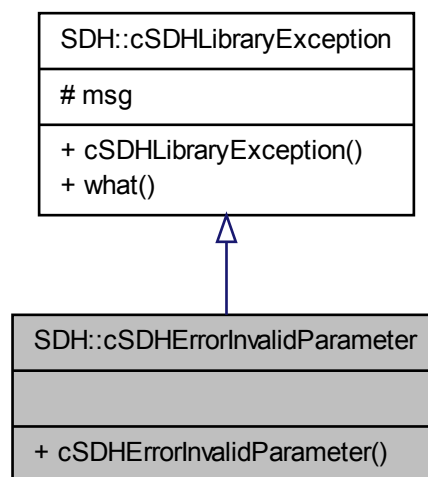
- [sdh/sdhexception.h](#)

10.23 SDH::cSDHErrorInvalidParameter Class Reference

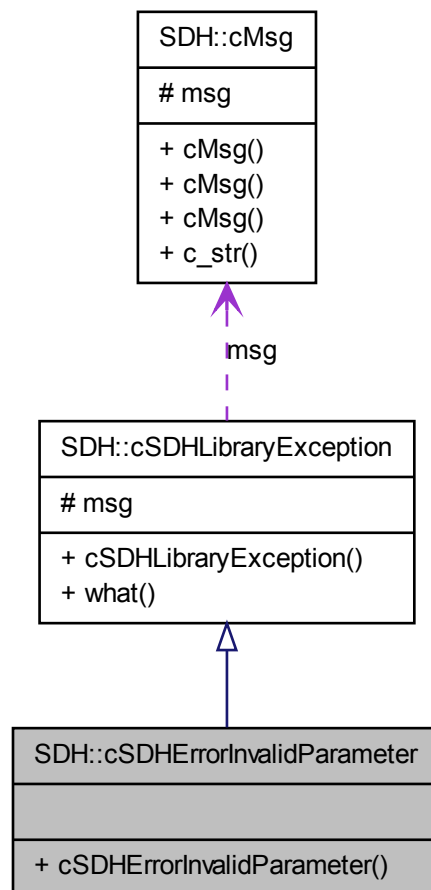
Derived exception class for exceptions related to invalid parameters.

```
#include <sdhbase.h>
```

Inheritance diagram for SDH::cSDHErrorInvalidParameter:



Collaboration diagram for SDH::cSDHErrorInvalidParameter:



Public Member Functions

- [cSDHErrorInvalidParameter](#) (cMsg const &_msg)

10.23.1 Detailed Description

Derived exception class for exceptions related to invalid parameters.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 SDH::cSDHErrorInvalidParameter::cSDHErrorInvalidParameter (cMsg const & _msg) [inline]

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

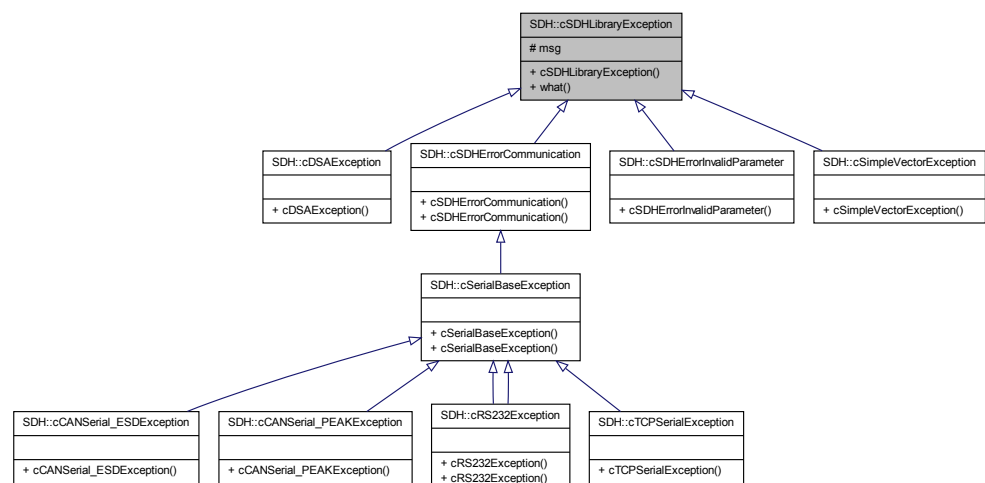
- [sdh/sdhbase.h](#)

10.24 SDH::cSDHLibraryException Class Reference

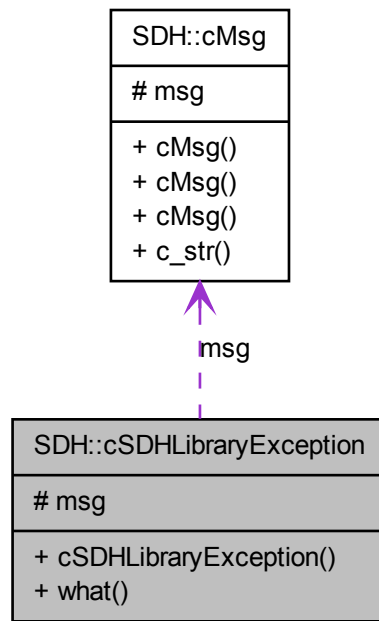
Base class for exceptions in the SDHLibrary-CPP.

```
#include <sdhexception.h>
```

Inheritance diagram for SDH::cSDHLibraryException:



Collaboration diagram for SDH::cSDHLibraryException:



Public Member Functions

- `cSDHLibraryException` (char const *_type, `cMsg` const &_msg)
- virtual const char * `what` () const throw ()

Protected Attributes

- `cMsg msg`

The message object.

10.24.1 Detailed Description

Base class for exceptions in the SDHLibrary-CPP. At construction time a `cMsg` object is stored in the `msg` member of the `cSDHLibraryException` object. The `cMsg` object should contain a string which further describes the actual cause of the exception thrown.

The string in the [cMsg](#) object can be queried with the overloaded [what\(\)](#) member function, just like in the `std::exception` class.

See the verbose description of the constructor [SDH::cSDHLibraryException::cSDHLibraryException\(\)](#) for exemplary use.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 cSDHLibraryException::cSDHLibraryException (char const * _type, cMsg const & _msg)

Constructor of sdh exception base class.

Parameters

<code>_type</code>	- the type name of the exception. By convention this is the class name of the exception
<code>_msg</code>	- a reference to a cMsg object that further describes the exception.

Remarks

- The `_type` parameter is mainly usefull in derived classes
- The `_msg` given as parameter is copied to the [msg](#) member. Thus the given `_msg` object can be an anonymous object, like in:

```
...
if ( v > v_max )
    throw new cSDHLibraryException( "cSDHLibraryException", cMsg( "Failed since v is invalid (v=%d > %d=v_max)", v, v_max ) );
```

- But exceptions of the base will hardly ever be thrown. Instead objects of derived, more specific classes will be thrown. This looks like:

```
// Derived exception class for more specific exceptions:
class VCC_EXPORT cDerivedException : public cSDHLibraryException
{
public:
    cDerivedException( cMsg const & _msg )
        : cSDHLibraryException( "cDerivedException", _msg )
    {}
}
// (Yes that is really all that must be done here!)

...

try
{
    ...
    if ( v > v_max )
        throw new cDerivedException( cMsg( "Failed since v is invalid (v=%d > %d=v_max)", v, v_max ) );
    ...
}
catch ( cDerivedException *e )
{
    cerr << "Caught exception " << e->what() << "\n";
}
```



```

        // handle exception
        ...

        // finally delete the caught exception
        delete e;
    }

```

10.24.3 Member Function Documentation

10.24.3.1 `const char * cSDHLibraryException::what () const throw ()` [virtual]

Return the [msg](#) member

10.24.4 Member Data Documentation

10.24.4.1 `cMsg SDH::cSDHLibraryException::msg` [protected]

The message object.

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

- [sdh/sdhexception.h](#)
- [sdh/sdhexception.cpp](#)

10.25 cSDHOptions Class Reference

class for command line option parsing holding option parsing results

```
#include <sdhoptions.h>
```

Public Member Functions

- [cSDHOptions](#) (char const *option_selection="general sdhcom_serial sdhcom_common sdhcom_esdcan sdhcom_peakcan sdhcom_cancommon sdhcom_tcp")
- [~cSDHOptions](#) ()
destructor, clean up
- int [Parse](#) (int argc, char **argv, char const *helptext, char const *progname, char const *version, char const *libname, char const *librelease)
- void [OpenCommunication](#) (NS_SDH cSDH &hand)

Public Attributes

- std::string [usage](#)
- int [debug_level](#)
- std::ostream * [debuglog](#)

- int [sdhport](#)
- char [sdh_rs_device](#) [MAX_DEV_LENGTH]
- double [timeout](#)
- unsigned long [rs232_baudrate](#)
- bool [use_can_esd](#)
- int [net](#)
- bool [use_can_peak](#)
- char [sdh_canpeak_device](#) [MAX_DEV_LENGTH]
- unsigned long [can_baudrate](#)
- unsigned int [id_read](#)
- unsigned int [id_write](#)
- bool [use_radians](#)
- bool [use_fahrenheit](#)
- double [period](#)
- int [dsaport](#)
- char [dsa_rs_device](#) [MAX_DEV_LENGTH]
- bool [do_RLE](#)
- int [framerate](#)
- bool [fullframe](#)
- bool [sensorinfo](#)
- bool [controllerinfo](#)
- int [matrixinfo](#) [6]
- double [sensitivity](#) [6]
- unsigned int [threshold](#) [6]
- bool [reset_to_default](#)
- bool [persistent](#)
- bool [showdsasettings](#)
- bool [use_tcp](#)
- std::string [tcp_adr](#)
- int [tcp_port](#)
- bool [dsa_use_tcp](#)
- int [dsa_tcp_port](#)

Static Public Attributes

- static int const [MAX_DEV_LENGTH](#) = 32

10.25.1 Detailed Description

class for command line option parsing holding option parsing results

10.25.2 Constructor & Destructor Documentation

10.25.2.1 `cSDHOptions::cSDHOptions (char const * option_selection = "general sdhcom_serial sdhcom_common sdhcom_esdcan sdhcom_peakcan sdhcom_cancommon sdhcom_tcp")`

constructor: init members to their default values

Parameters

<i>option_selection</i>	<p>- string that names the options to include in helptext for online help. With a text including one of the following keywords the corresponding helptext is added to the usage helptext</p> <ul style="list-style-type: none"> • "general" see sdhusage_general • "sdhcom_serial" see sdhusage_sdhcom_serial • "sdhcom_common" see sdhusage_sdhcom_common • "sdhcom_esdcan" see sdhusage_sdhcom_esdcan • "sdhcom_peakcan" see sdhusage_sdhcom_peakcan • "sdhcom_cancommon" see sdhusage_sdhcom_cancommon • "sdhcom_tcp" see sdhusage_sdhcom_tcp • "sdhoother" see sdhusage_sdhoother • "dsacom" see sdhusage_dsacom • "dsaoother" see sdhusage_dsaoother
-------------------------	---

10.25.2.2 `cSDHOptions::~~cSDHOptions ()`

destructor, clean up

10.25.3 Member Function Documentation

10.25.3.1 `void cSDHOptions::OpenCommunication (NS_SDH cSDH & hand)`

convenience function to open the communication of the given *hand* object according to the parsed parameters.

Parameters

<i>hand</i>	- reference to a cSDH object to open
-------------	--------------------------------------

10.25.3.2 `int cSDHOptions::Parse (int argc, char ** argv, char const * helptext, char const * progrname, char const * version, char const * libname, char const * librelease)`

parse the command line parameters *argc*, *argv* into members. *helptext*, *progrname*, *version*, *libname* and *librelease* are used when printing online help. start parsing at option with index **p_option_index* parse all options if *parse_all* is true, else only one option is parsed

Returns

the optind index of the first non option argument in *argv*

10.25.4 Member Data Documentation

10.25.4.1 unsigned long cSDHOptions::can_baudrate

10.25.4.2 bool cSDHOptions::controllerinfo

10.25.4.3 int cSDHOptions::debug_level

10.25.4.4 std::ostream* cSDHOptions::debuglog

10.25.4.5 bool cSDHOptions::do_RLE

10.25.4.6 char cSDHOptions::dsa_rs_device[MAX_DEV_LENGTH]

10.25.4.7 int cSDHOptions::dsa_tcp_port

10.25.4.8 bool cSDHOptions::dsa_use_tcp

10.25.4.9 int cSDHOptions::dsaport

10.25.4.10 int cSDHOptions::framerate

10.25.4.11 bool cSDHOptions::fullframe

10.25.4.12 unsigned int cSDHOptions::id_read

10.25.4.13 unsigned int cSDHOptions::id_write

10.25.4.14 int cSDHOptions::matrixinfo[6]

10.25.4.15 int const cSDHOptions::MAX_DEV_LENGTH = 32 [static]

10.25.4.16 int cSDHOptions::net

10.25.4.17 double cSDHOptions::period

10.25.4.18 bool cSDHOptions::persistent

10.25.4.19 bool cSDHOptions::reset_to_default

10.25.4.20 unsigned long cSDHOptions::rs232_baudrate

10.25.4.21 char cSDHOptions::sdh_canpeak_device[MAX_DEV_LENGTH]

10.25.4.22 char cSDHOptions::sdh_rs_device[MAX_DEV_LENGTH]

10.25.4.23 int cSDHOptions::sdhport

10.25.4.24 double cSDHOptions::sensitivity[6]

10.25.4.25 bool cSDHOptions::sensorinfo

10.25.4.26 bool cSDHOptions::showdsasettings

10.25.4.27 std::string cSDHOptions::tcp_adr

10.25.4.28 int cSDHOptions::tcp_port

10.25.4.29 unsigned int cSDHOptions::threshold[6]

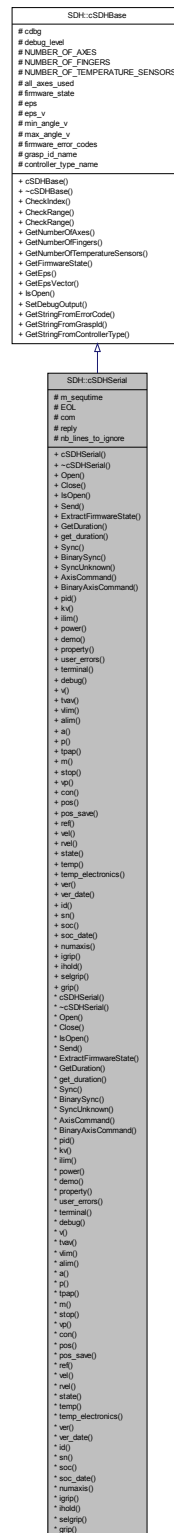
- [demo/sdhoptions.h](#)
- [demo/sdhoptions.cpp](#)

10.26 SDH::cSDHSerial Class Reference

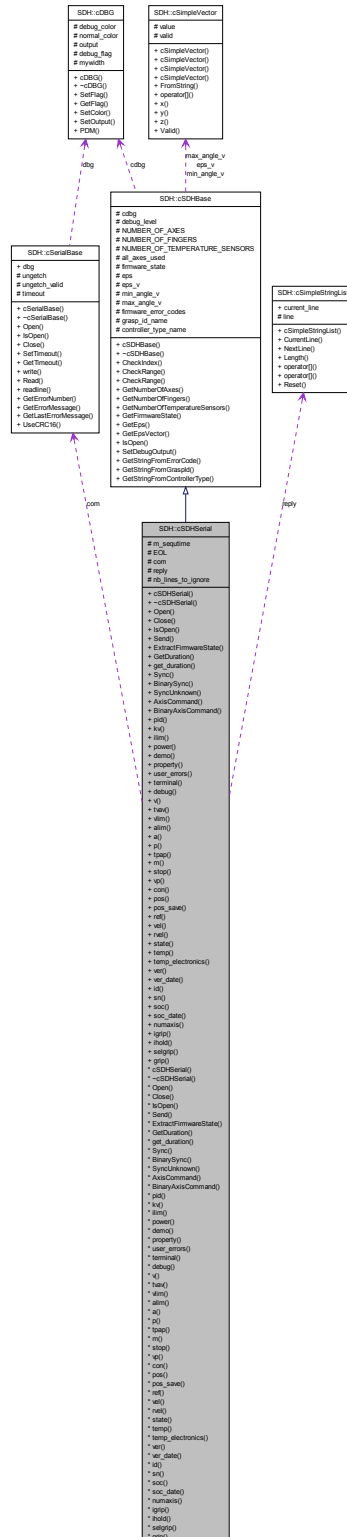
The class to communicate with a SDH via RS232.

```
#include <sdhserial.h>
```

Inheritance diagram for SDH::cSDHSerial:



Collaboration diagram for SDH::cSDHSerial:



Public Member Functions

Internal methods

- [cSDHSerial](#) (int _debug_level=0)
Constructor of [cSDHSerial](#).
- virtual [~cSDHSerial](#) ()
- void [Open](#) ([cSerialBase](#) *_com) throw ([cSDHLibraryException](#)*)
- void [Close](#) () throw ([cSDHLibraryException](#)*)
- virtual bool [IsOpen](#) (void)
- void [Send](#) (char const *s, int nb_lines=All, int nb_lines_total=All, int max_retries=3) throw ([cSDHLibraryException](#)*)
- void [ExtractFirmwareState](#) () throw ([cSDHErrorCommunication](#)*)
- double [GetDuration](#) (char *line) throw ([cSDHErrorCommunication](#)*)
- double [get_duration](#) (void)
- void [Sync](#) () throw ([cSDHErrorCommunication](#)*)
- void [BinarySync](#) (double timeout_s=0.5) throw ([cSDHErrorCommunication](#)*)
- void [SyncUnknown](#) () throw ([cSDHErrorCommunication](#)*)
- [cSimpleVector](#) [AxisCommand](#) (char const *command, int axis=All, double *value=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [BinaryAxisCommand](#) ([eCommandCode](#) command, int axis=All, double *value=NULL) throw ([cSDHLibraryException](#)*)

Setup and configuration methods

- [cSimpleVector](#) [pid](#) (int axis, double *p=NULL, double *i=NULL, double *d=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [kv](#) (int axis=All, double *kv=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [ilim](#) (int axis=All, double *limit=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [power](#) (int axis=All, double *flag=NULL) throw ([cSDHLibraryException](#)*)

Misc. methods

- void [demo](#) (bool onoff)
- int [property](#) (char const *propname, int value)
- int [user_errors](#) (int value)
- int [terminal](#) (int value)
- int [debug](#) (int value)

Movement methods

- [cSimpleVector](#) [v](#) (int axis=All, double *velocity=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [tvav](#) (int axis=All, double *velocity=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [vlim](#) (int axis=All, double *dummy=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [alim](#) (int axis=All, double *dummy=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [a](#) (int axis=All, double *acceleration=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [p](#) (int axis=All, double *angle=NULL) throw ([cSDHLibraryException](#)*)
- [cSimpleVector](#) [tpap](#) (int axis=All, double *angle=NULL) throw ([cSDHLibraryException](#)*)
- double [m](#) (bool sequ) throw ([cSDHLibraryException](#)*)
- void [stop](#) (void) throw ([cSDHLibraryException](#)*)

- [eVelocityProfile vp](#) ([eVelocityProfile](#) velocity_profile=[eVP_INVALID](#)) throw ([cSDHLibraryException*](#))
- [eControllerType con](#) ([eControllerType](#) controller) throw ([cSDHLibraryException*](#))

Diagnostic and identification methods

- [cSimpleVector pos](#) (int axis=All, double *dummy=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector pos_save](#) (int axis=All, double *value=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector ref](#) (int axis=All, double *value=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector vel](#) (int axis=All, double *dummy=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector rvel](#) (int axis, double *dummy=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector state](#) (int axis=All, double *dummy=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector temp](#) (void) throw ([cSDHLibraryException*](#))
- [cSimpleVector temp_electronics](#) (void) throw ([cSDHLibraryException*](#))
- char * [ver](#) (void) throw ([cSDHLibraryException*](#))
- char * [ver_date](#) (void) throw ([cSDHLibraryException*](#))
- char * [id](#) (void) throw ([cSDHLibraryException*](#))
- char * [sn](#) (void) throw ([cSDHLibraryException*](#))
- char * [soc](#) (void) throw ([cSDHLibraryException*](#))
- char * [soc_date](#) (void) throw ([cSDHLibraryException*](#))
- int [numaxis](#) (void) throw ([cSDHLibraryException*](#))

Grip methods

- [cSimpleVector igrip](#) (int axis=All, double *limit=NULL) throw ([cSDHLibraryException*](#))
- [cSimpleVector ihold](#) (int axis=All, double *limit=NULL) throw ([cSDHLibraryException*](#))
- double [selgrip](#) ([eGraspId](#) grip, bool sequ) throw ([cSDHLibraryException*](#))
- double [grip](#) (double close, double velocity, bool sequ) throw ([cSDHLibraryException*](#))

Protected Attributes

- double [m_sequetime](#)
additional time in seconds to wait for sequential execution of m command (as these are always executed non-sequentially by the SDH firmware)
- char const * [EOL](#)
String to use as "End Of Line" marker when sending to SDH.
- [cSerialBase](#) * [com](#)
The communication object to the serial device (RS232 port or ESD CAN net)
- [cSimpleStringList](#) [reply](#)
Space for the replies from the SDH.
- int [nb_lines_to_ignore](#)
number of remaining reply lines of a previous (non-sequential) command

Friends

- struct [sSDHBinaryRequest](#)
- struct [sSDHBinaryResponse](#)

10.26.1 Detailed Description

The class to communicate with a SDH via RS232. End-Users should **NOT** use this class directly! The interface of [cSDHSerial](#) is subject to change in future releases. End users should use the class [cSDH](#) instead, as that interface is considered more stable.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 [cSDHSerial::cSDHSerial](#) (int *_debug_level* = 0)

Constructor of [cSDHSerial](#).

Parameters

<i>_debug_level</i>	: debug level of the created object. If the <i>debug_level</i> of an object is > 0 then it will output debug messages. (forwarded to constructor of base class)
---------------------	---

String to use as "End Of Line" marker when sending to SDH

10.26.2.2 [virtual SDH::cSDHSerial::~~cSDHSerial](#) () [inline, virtual]

virtual destructor to make compiler happy

10.26.3 Member Function Documentation

10.26.3.1 [cSimpleVector cSDHSerial::a](#) (int *axis* = All, double * *acceleration* = NULL) throw (cSDHLibraryException*)

Get/Set target acceleration for axis. (NOT the actual acceleration!)

- If axis is All and acceleration is None then a NUMBER_OF_AXES-list of the actually set target accelerations is returned
- If axis is a single number and acceleration is None then the target acceleration for that axis is returned.
- If axis and acceleration are single numbers then the target acceleration for that axis is set (and returned).
- If axis is All and acceleration is a NUMBER_OF_AXES-vector then all axes target accelerations are set accordingly, the NUMBER_OF_AXES-list is returned.

Accelerations are set/reported in degrees per second squared.

10.26.3.2 cSimpleVector cSDHSerial::alim (int *axis* = All, double * *dummy* = NULL) throw (cSDHLibraryException*)

Get acceleration limits.

- If *axis* is All then a NUMBER_OF_AXES-list of the acceleration limits is returned
- If *axis* is a single number then the acceleration limit for that axis is returned.

dummy parameter is just needed to make this function have the same signature as e.g. [v\(\)](#), so it can be used as a function pointer.

Acceleration limits are reported in degrees per (second*second).

10.26.3.3 cSimpleVector cSDHSerial::AxisCommand (char const * *command*, int *axis* = All, double * *value* = NULL) throw (cSDHLibraryException*)

Get/Set values.

- If *axis* is All and *value* is None then a NUMBER_OF_AXES-list of the actual values read from the SDH is returned
- If *axis* is a single number and *value* is None then the actual value for that axis is read from the SDH and is returned
- If *axis* and *value* are single numbers then that value is set for that axis and returned.
- If *axis* is All and *value* is a NUMBER_OF_AXES-vector then all axes values are set accordingly, a NUMBER_OF_AXES-list is returned.

10.26.3.4 cSimpleVector cSDHSerial::BinaryAxisCommand (eCommandCode *command*, int *axis* = All, double * *value* = NULL) throw (cSDHLibraryException*)

Get/Set values using binary mode.

- If *axis* is All and *value* is None then a NUMBER_OF_AXES-list of the actual values read from the SDH is returned
- If *axis* is a single number and *value* is None then the actual value for that axis is read from the SDH and is returned
- If *axis* and *value* are single numbers then that value is set for that axis and returned.
- If *axis* is All and *value* is a NUMBER_OF_AXES-vector then all axes values are set accordingly, a NUMBER_OF_AXES-list is returned.

10.26.3.5 void cSDHSerial::BinarySync (double *timeout_s* = 0.5) throw (cSDHErrorCommunication*)

Read all available bytes within *timeout_s* seconds to resync execution of PC and SDH.

10.26.3.6 void cSDHSerial::Close (void) throw (cSDHLibraryException*)

Close connection to serial port.

10.26.3.7 cSDHBase::eControllerType cSDHSerial::con (eControllerType *controller*) throw (cSDHLibraryException*)

Get/set controller type.

If *controller* is < 0 then the actually set controller is read from the SDH firmware and returned. Else the given controller type is set in the SDH firmware if valid.

10.26.3.8 int cSDHSerial::debug (int *value*)

10.26.3.9 void cSDHSerial::demo (bool *onoff*)

Enable/disable SCHUNK demo

10.26.3.10 void cSDHSerial::ExtractFirmwareState () throw (cSDHErrorCommunication*)

Try to extract the state of the SDH firmware from the last reply

10.26.3.11 double cSDHSerial::get_duration (void)

Send *get_duration* command. Returns the calculated duration of the currently configured movement (target positions, velocities, accelerations and velocity profile.

return the expected duration of the execution of the command in seconds

10.26.3.12 double cSDHSerial::GetDuration (char * *line*) throw (cSDHErrorCommunication*)

Return duration of the execution of a SDH command as reported by line

10.26.3.13 double cSDHSerial::grip (double *close*, double *velocity*, bool *sequ*) throw (cSDHLibraryException*)

send "grip=close,velocity" command to SDH *close* : [0.0 .. 1.0] where 0.0 is 'fully opened' and 1.0 is 'fully closed' *velocity* :]0.0 .. 100.0] where 0.0 (not allowed) is

very slow and 100.0 is very fast

If sequ is True then wait until SDH hardware fully executed the command. Else return immediately and do not wait until SDH hardware fully executed the command.

This seems to work with sin square velocity profile only, so the velocity profile is switched to that if necessary.

return the expected duration of the execution of the command in seconds

10.26.3.14 `char * cSDHSerial::id (void) throw (cSDHLibraryException*)`

Return id of SDH

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.15 `cSimpleVector cSDHSerial::igrip (int axis = All, double * limit = NULL) throw (cSDHLibraryException*)`

Get/Set motor current limits for grip commands

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the actually set motor current limits is returned
- If axis is a single number and limit is None then the motor current limit for that axis is returned.
- If axis and limit are single numbers then the motor current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes motor current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.26.3.16 `cSimpleVector cSDHSerial::ihold (int axis = All, double * limit = NULL) throw (cSDHLibraryException*)`

Get/Set motor current limits for hold commands

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the actually set motor current limits is returned
- If axis is a single number and limit is None then the motor current limit for that axis is returned.
- If axis and limit are single numbers then the motor current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes motor current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.26.3.17 `cSimpleVector cSDHSerial::lim (int axis = All, double * limit = NULL) throw (cSDHLibraryException*)`

Get/Set actual motor current limit for m command

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the actually set motor current limits is returned
- If axis is a single number and limit is None then the motor current limit for that axis is returned.
- If axis and limit are single numbers then the motor current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes motor current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.26.3.18 `bool cSDHSerial::isOpen (void) [virtual]`

Return true if connection to SDH firmware/hardware is open

Implements [SDH::cSDHBase](#).

10.26.3.19 `cSimpleVector cSDHSerial::kv (int axis = All, double * kv = NULL) throw (cSDHLibraryException*)`

Get/Set kv parameter

- If axis is All and kv is None then a NUMBER_OF_AXES-list of the actually set kv parameters is returned
- If axis is a single number and kv is None then the kv parameter for that axis is returned.
- If axis and kv are single numbers then the kv parameter for that axis is set (and returned).
- If axis is All and kv is a NUMBER_OF_AXES-vector then all axes kv parameters are set accordingly, NUMBER_OF_AXES-list is returned.

Bug

With [SDH](#) firmware 0.0.2.9 [kv\(\)](#) might not respond kv value correctly in case it was changed before. With [SDH](#) firmwares 0.0.2.10 and newer this now works.
=> **Resolved in [SDH](#) firmware 0.0.2.10**

10.26.3.20 double cSDHSerial::m (bool sequ) throw (cSDHLibraryException*)

Send move command. Moves all enabled axes to their previously set target angle. The movement duration is determined by that axis that takes longest with its actually set velocity. The actual velocity of all other axes is set so that all axes begin and end their movements synchronously.

If sequ is True then wait until SDH hardware fully executed the command. Else return immediately and do not wait until SDH hardware fully executed the command.

return the expected duration of the execution of the command in seconds

10.26.3.21 int cSDHSerial::numaxis (void) throw (cSDHLibraryException*)

Return number of axis of SDH

10.26.3.22 void cSDHSerial::Open (cSerialBase * _com) throw (cSDHLibraryException*)

Open the serial device and check connection to SDH by querying the SDH firmware version

Parameters

<code>_com</code>	- ptr to the serial device to use
-------------------	-----------------------------------

This may throw an exception on failure.

The serial port on the PC-side can be opened successfully even if no SDH is attached. Therefore this routine tries to read the SDH firmware version with a 1s timeout after the port is opened. If the SDH does not reply in time then

- an error message is printed on stderr,
- the port is closed
- and a cSerialBaseException* exception is thrown.

Bug

SCHUNK-internal bugzilla ID: Bug 1517

With [SDH](#) firmware 0.0.3.x the first connection to a newly powered up [SDH](#) can yield an error especially when connecting via TCP.

=> **Resolved in SDHLibrary-C++ 0.0.2.10**

10.26.3.23 cSimpleVector cSDHSerial::p (int axis = All, double * angle = NULL) throw (cSDHLibraryException*)

Get/Set target angle for axis. (NOT the actual angle!)

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the actually set target angles is returned
- If axis is a single number and angle is None then the target angle for that axis is returned.
- If axis and angle are single numbers then the target angle for that axis is set (and returned).
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes target angles are set accordingly, the NUMBER_OF_AXES-list is returned.

Angles are set/reported in degrees.

10.26.3.24 `cSimpleVector cSDHSerial::pid (int axis, double * p = NULL, double * i = NULL, double * d = NULL) throw (cSDHLibraryException*)`

Get/Set PID controller parameters

- axis must be a single number: the index of the axis to get/set
- If p,i,d are None then a list of the actually set PID controller parameters of the axis is returned
- If p,i,d are numbers then the PID controller parameters for that axis are set (and returned).

Bug

With [SDH](#) firmware 0.0.2.9 `pid()` might not respond pid values correctly in case these were changed before. With [SDH](#) firmwares 0.0.2.10 and newer this now works.

=> **Resolved in [SDH](#) firmware 0.0.2.10**

10.26.3.25 `cSimpleVector cSDHSerial::pos (int axis = All, double * dummy = NULL) throw (cSDHLibraryException*)`

Get actual angle/s of axis/axes.

- If axis is All then a NUMBER_OF_AXES-vector of the actual axis angles is returned
- If axis is a single number then the actual angle of that axis is returned.

Angles are reported in degrees.

Remarks

dummy ptr is never used, but needed nonetheless to make the signature of the function the same as for the other axis-access functions. This way a pointer to it can be used as a pointer to the other functions, which is needed by the generic `cSDH::SetAxisValue` and `cSDH::GetAxisValue` functions.

**10.26.3.26 cSimpleVector cSDHSerial::pos_save (int *axis* = All, double * *value* = NULL)
throw (cSDHLibraryException*)**

Save actual angle/s to non volatile memory. (Usefull for axes that dont have an absolute encoder)

- If value is None then an exception is thrown since this is NOT usefull if any axis has an absolute encoder that the LLC knows about since these positions will be invalidated at the next start
- If axis and value are single numbers then that axis is saved.
- If axis is All and value is a NUMBER_OF_AXES-vector then all axes are saved if the corresponding value is 1.
- This will yield a E_RANGE_ERROR if any of the given values is not 0 or 1

**10.26.3.27 cSimpleVector cSDHSerial::power (int *axis* = All, double * *flag* = NULL)
throw (cSDHLibraryException*)**

Get/Set actual power state

- If axis is All and flag is None then a NUMBER_OF_AXES-list of the actually set power states is returned
- If axis is a single number and flag is None then the power state for that axis is returned.
- If axis is a single number and flag is a single number or a boolean value then the power state for that axis is set (and returned).
- If axis is All and flag is a NUMBER_OF_AXES-vector then all axes power states are set accordingly, the NUMBER_OF_AXES-list is returned.
- If axis is All and flag is a a single number or a boolean value then all axes power states are set to that value, the NUMBER_OF_AXES-list is returned.

10.26.3.28 int cSDHSerial::property (char const * *propname*, int *value*)

Set named property

Valid propnames are:

- "user_errors"
- "terminal"
- "debug"

10.26.3.29 `cSimpleVector cSDHSerial::ref (int axis = ALL, double * value = NULL) throw (cSDHLibraryException*)`

Do reference movements with selected axes. (Usefull for axes that dont have an absolute encoder)

each *value* must be either

- 0 : do not reference
- 1 : reference till mechanical block in positive direction
- 2 : reference till mechanical block in negative direction
- If *axis* and *value* are single numbers then that axis is referenced as requested
- If *axis* is All and *value* is a NUMBER_OF_AXES-vector then all axes are referenced as requested.
- This will yield a E_RANGE_ERROR if any of the given values is not 0 or 1 or 2

10.26.3.30 `cSimpleVector cSDHSerial::rvel (int axis, double * dummy = NULL) throw (cSDHLibraryException*)`

Get current reference angular velocity/s of axis/axes. The reference angular velocity is used by the eCT_VELOCITY_ACCELERATION controller type only.

- If *axis* is All then a NUMBER_OF_AXES-vector of the current reference velocities is returned
- If *axis* is a single number then the current reference angular velocity of that axis is returned.

Angular velocities are reported in degrees/second.

Remarks

- *dummy* ptr is never used, but needed nonetheless to make the signature of the function the same as for the other axis-access functions. This way a pointer to it can be used as a pointer to the other functions, which is needed by the generic `cSDH::SetAxisValue` and `cSDH::GetAxisValue` functions.
- the underlying rvel command of the SDH firmware is not available in SDH firmwares prior to 0.0.2.6. For such hands calling rvel will fail miserably.

10.26.3.31 `double cSDHSerial::selgrip (eGraspId grip, bool sequ) throw (cSDHLibraryException*)`

Send "selgrip grip" command to SDH. Where grip is in [0..eGID_DIMENSION-1] or one of the eGraspId enums.

If sequ is True then wait until SDH hardware fully executed the command. Else return immediately and do not wait until SDH hardware fully executed the command.

This seems to work with sin square velocity profile only, so the velocity profile is switched to that if necessary.

return the expected duration of the execution of the command in seconds

10.26.3.32 `void cSDHSerial::Send (char const * s, int nb_lines = All, int nb_lines_total = All, int max_retries = 3) throw (cSDHLibraryException*)`

Send command string s+EOL to com and read reply according to nb_lines.

If nb_lines == All then reply lines are read until a line without "@" prefix is found. If nb_lines != All it is the number of lines to read.

firmware_state is set according to reply (if read) nb_lines_total contains the total number of lines replied for the s command. If fewer lines are read then nb_lines_total-nb_lines will be remembered to be ignored before the next command can be sent.

Return a list of all read lines of the reply from the SDH hardware.

10.26.3.33 `char * cSDHSerial::sn (void) throw (cSDHLibraryException*)`

Return sn of SDH

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.34 `char * cSDHSerial::soc (void) throw (cSDHLibraryException*)`

Return soc (System On Chip) ID of SDH

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.35 `char * cSDHSerial::soc_date (void) throw (cSDHLibraryException*)`

Return date of soc (System On Chip) ID of SDH

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.36 `cSimpleVector cSDHSerial::state (int axis = All, double * dummy = NULL)
throw (cSDHLibraryException*)`

Get actual state/s of axis/axes.

A state of 0 means "not moving" while 1 means "moving".

- If axis is All then a NUMBER_OF_AXES-vector of the actual axis states is returned
- If axis is a single number then the actual state of that axis is returned.

10.26.3.37 `void cSDHSerial::stop (void) throw (cSDHLibraryException*)`

Stop sdh.

Will NOT interrupt a previous "selgrip" or "grip" command, only an "m" command!

10.26.3.38 `void cSDHSerial::Sync () throw (cSDHErrorCommunication*)`

Read all pending lines from SDH to resync execution of PC and SDH.

10.26.3.39 `void cSDHSerial::SyncUnknown () throw (cSDHErrorCommunication*)`

Read an unknown number of lines from SDH to resync execution of PC and SDH.

10.26.3.40 `cSimpleVector cSDHSerial::temp (void) throw (cSDHLibraryException*)`

Get actual temperatures of the axis motors

Returns a list of the actual controller and driver temperature in degrees celsius.

10.26.3.41 `cSimpleVector cSDHSerial::temp_electronics (void) throw
(cSDHLibraryException*)`

Get actual temperatures of the electronics, i.e. the FPGA and the PCB. (FPGA = Field Programmable Gate Array = the reprogrammable chip with the soft processors) (PCB = Printed Circuit Board)

Returns a list of the actual controller and driver temperature in degrees celsius.

10.26.3.42 `int cSDHSerial::terminal (int value)`

10.26.3.43 `cSimpleVector cSDHSerial::tpap (int axis = All, double * angle = NULL)
throw (cSDHLibraryException*)`

Set target angle and get actual angle for axis.

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the actual angles is returned
- If axis is a single number and angle is None then the actual angle for that axis is returned.
- If axis and angle are single numbers then the target angle for that axis is set and the actual angles are returned.
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes target angles are set accordingly, the NUMBER_OF_AXES-list of actual angles is returned.

Angles are set/reported in degrees.

10.26.3.44 `cSimpleVector cSDHSerial::tvav (int axis = All, double * velocity = NULL)
throw (cSDHLibraryException*)`

Set target velocity and get actual velocity!)

The default velocity is 40 deg/s for axes 0-6 in eCT_POSE controller type. The default velocity is 0.0 deg/s for axes 0-6 in eCT_VELOCITY and eCT_VELOCITY-ACCELERATION controller types.

- If axis is All and velocity is None then a NUMBER_OF_AXES-list of the actual velocities is returned
- If axis is a single number and velocity is None then the actual velocity for that axis is returned.
- If axis and velocity are single numbers then the target velocity for that axis is set and the actual velocity is returned.
- If axis is All and velocity is a NUMBER_OF_AXES-vector then all axes target velocities are set accordingly, the NUMBER_OF_AXES-list of actual velocities is returned.

Velocities are set/reported in degrees per second.

Bug

With SDH firmware < 0.0.1.0 axis 0 can go no faster than 14 deg/s
=> **Resolved in SDH firmware 0.0.1.0**

10.26.3.45 `int cSDHSerial::user_errors (int value)`

10.26.3.46 `cSimpleVector cSDHSerial::v (int axis = All, double * velocity = NULL) throw (cSDHLibraryException*)`

Get/Set target velocity. (NOT the actual velocity!)

The default velocity is 40 deg/s for axes 0-6 in eCT_POSE controller type. The default velocity is 0.0 deg/s for axes 0-6 in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller types.

- If axis is All and velocity is None then a NUMBER_OF_AXES-list of the actually set target velocities is returned
- If axis is a single number and velocity is None then the target velocity for that axis is returned.
- If axis and velocity are single numbers then the target velocity for that axis is set (and returned).
- If axis is All and velocity is a NUMBER_OF_AXES-vector then all axes target velocities are set accordingly, the NUMBER_OF_AXES-list is returned.

Velocities are set/reported in degrees per second.

Bug

With SDH firmware < 0.0.1.0 axis 0 can go no faster than 14 deg/s
=> **Resolved in SDH firmware 0.0.1.0**

10.26.3.47 `cSimpleVector cSDHSerial::vel (int axis = All, double * dummy = NULL) throw (cSDHLibraryException*)`

Get actual angular velocity/s of axis/axes.

- If axis is All then a NUMBER_OF_AXES-vector of the actual axis angular velocities is returned
- If axis is a single number then the actual angular velocity of that axis is returned.

Angular velocities are reported in degrees/second.

Remarks

dummy ptr is never used, but needed nonetheless to make the signature of the function the same as for the other axis-access functions. This way a pointer to it can be used as a pointer to the other functions, which is needed by the generic cSDH::SetAxisValue and cSDH::GetAxisValue functions.

10.26.3.48 `char * cSDHSerial::ver (void) throw (cSDHLibraryException*)`

Return version of SDH firmware

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.49 `char * cSDHSerial::ver_date (void) throw (cSDHLibraryException*)`

Return date of SDH firmware

Attention

The string returned is stored internally in this object and might be overwritten by the next command to this object

10.26.3.50 `cSimpleVector cSDHSerial::vlim (int axis = All, double * dummy = NULL) throw (cSDHLibraryException*)`

Get velocity limits.

- If axis is All then a NUMBER_OF_AXES-list of the velocity limits is returned
- If axis is a single number then the velocity limit for that axis is returned.

dummy parameter is just needed to make this function have the same signature as e.g. [v\(\)](#), so it can be used as a function pointer.

Velocity limits are reported in degrees per second.

10.26.3.51 `cSDHBase::eVelocityProfile cSDHSerial::vp (eVelocityProfile velocity_profile = eVP_INVALID) throw (cSDHLibraryException*)`

Get/set velocity profile.

If *velocity_profile* is < 0 then the actually set velocity profile is read from the SDH firmware and returned. Else the given *velocity_profile* type is set in the SDH firmware if valid.

10.26.4 Friends And Related Function Documentation

10.26.4.1 `friend struct sSDHBinaryRequest` [friend]

10.26.4.2 `friend struct sSDHBinaryResponse` [friend]

10.26.5 Member Data Documentation

10.26.5.1 `cSerialBase* SDH::cSDHSerial::com` [protected]

The communication object to the serial device (RS232 port or ESD CAN net)

10.26.5.2 `char const* SDH::cSDHSerial::EOL` [protected]

String to use as "End Of Line" marker when sending to SDH.

10.26.5.3 `double SDH::cSDHSerial::m_sequetime` [protected]

additional time in seconds to wait for sequential execution of m command (as these are always executed non-sequentially by the SDH firmware)

10.26.5.4 `int SDH::cSDHSerial::nb_lines_to_ignore` [protected]

number of remaining reply lines of a previous (non-sequential) command

10.26.5.5 `cSimpleStringList SDH::cSDHSerial::reply` [protected]

Space for the replies from the SDH.

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

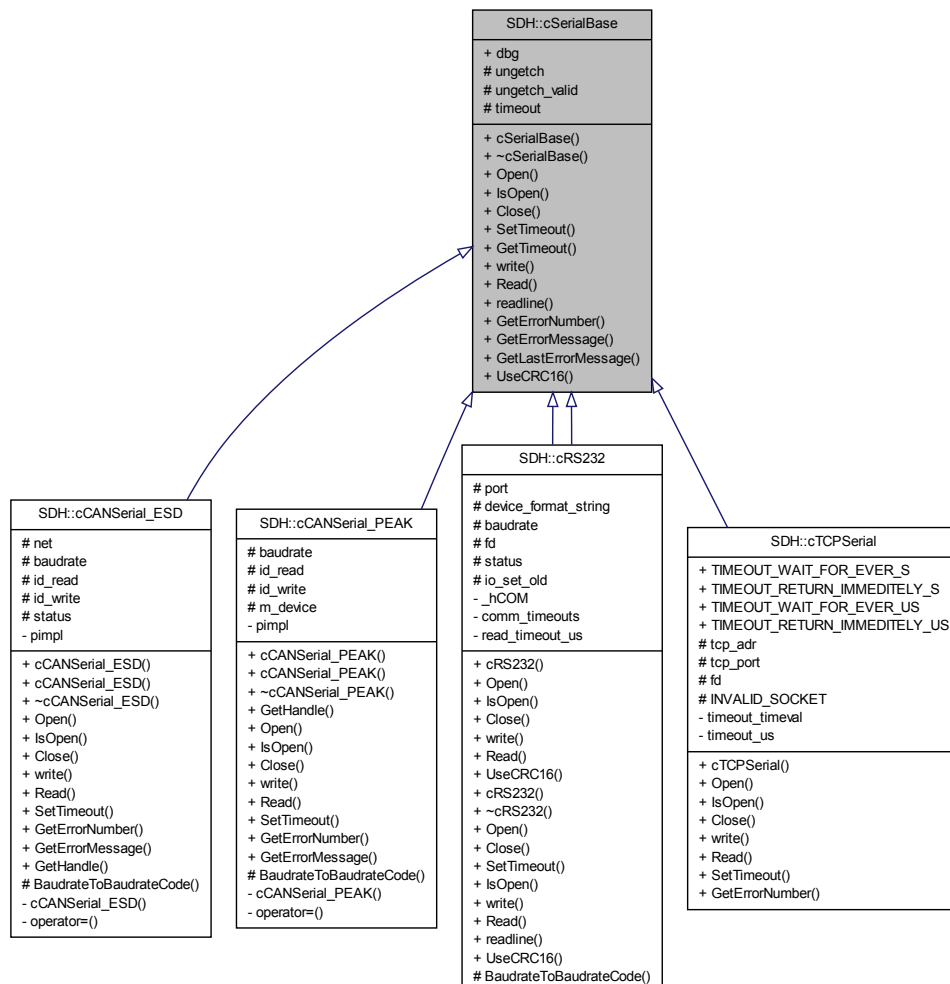
- [sdh/sdhserial.h](#)
- [sdh/sdhserial.cpp](#)

10.27 SDH::cSerialBase Class Reference

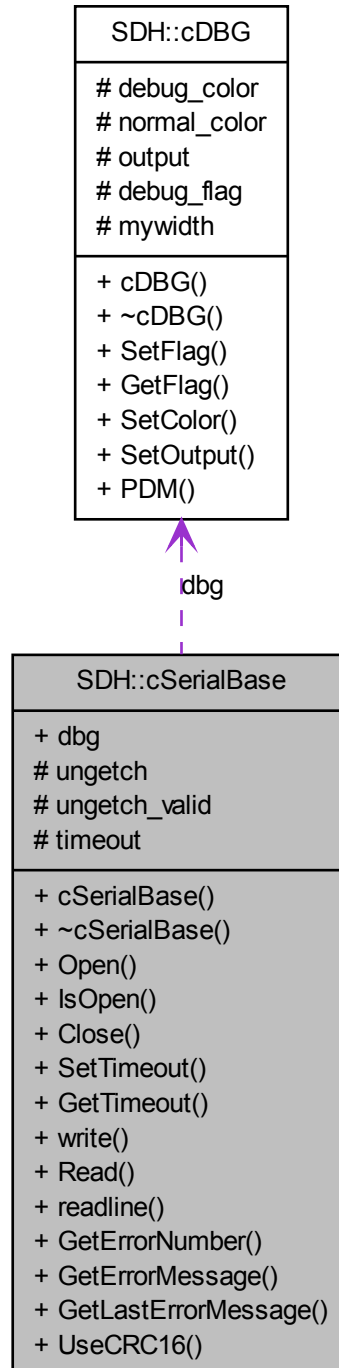
Low-level communication class to access a serial port.

```
#include <serialbase.h>
```

Inheritance diagram for SDH::cSerialBase:



Collaboration diagram for SDH::cSerialBase:



Classes

- class [cSetTimeoutTemporarily](#)
helper class to set timeout of `_serial_base` on construction and reset to previous value on destruction. (RAII-idiom)

Public Types

- typedef int [tErrorCode](#)
type of the error code, `DWORD` on windows and `int` on Linux/cygwin

Public Member Functions

- [cSerialBase](#) ()
ctor
- virtual [~cSerialBase](#) (void)
dtor
- virtual void [Open](#) (void)=0 throw (cSerialBaseException*)
Open rs232 port port.
- virtual bool [IsOpen](#) (void)=0 throw ()
Return true if communication channel is open.
- virtual void [Close](#) (void)=0 throw (cSerialBaseException*)
Close the previously opened communication channel.
- virtual void [SetTimeout](#) (double _timeout) throw (cSerialBaseException*)
set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- virtual double [GetTimeout](#) ()
get the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- virtual int [write](#) (char const *ptr, int len=0)=0 throw (cSerialBaseException*)
Write data to a previously opened port.
- virtual ssize_t [Read](#) (void *data, ssize_t [size](#), long timeout_us, bool return_on_less_data)=0 throw (cSerialBaseException*)
- virtual char * [readline](#) (char *line, int [size](#), char const *eol="\n", bool return_on_less_data=false) throw (cSerialBaseException*)
Read a line from the device.

- virtual `tErrorCode GetErrorNumber ()`
- virtual `char const * GetErrorMessage (tErrorCode dw)`
- `char const * GetLastErrorMessage (void)`
return the last error message as string. The string returned will be overwritten by the next call to the function
- virtual `bool UseCRC16 ()`

Public Attributes

- `cDBG dbg`
A stream object to print colored debug messages.

Protected Attributes

- `char ungetch`
an already read data byte of the next line
- `bool ungetch_valid`
Flag, true if ungetch is valid.
- `double timeout`
timeout in seconds

10.27.1 Detailed Description

Low-level communication class to access a serial port. (This is an abstract base class with pure virtual functions)

10.27.2 Member Typedef Documentation

10.27.2.1 `typedef int SDH::cSerialBase::tErrorCode`

type of the error code, `DWORD` on windows and `int` on Linux/cygwin

10.27.3 Constructor & Destructor Documentation

10.27.3.1 `SDH::cSerialBase::cSerialBase () [inline]`

ctor

10.27.3.2 virtual SDH::cSerialBase::~cSerialBase (void) [inline, virtual]

dtor

10.27.4 Member Function Documentation

10.27.4.1 virtual void SDH::cSerialBase::Close (void) throw (cSerialBaseException*)
[pure virtual]

Close the previously opened communication channel.

Implemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.4.2 char const * cSerialBase::GetErrorMessage (tErrorCode dw) [virtual]

Helper function that returns an error message for error code dw.

- On windows FormatMessageA() is used
- On cygwin/linux this uses errno

Remarks

The string returned will be overwritten by the next call to the function

Reimplemented in [SDH::cCANSerial_ESD](#), and [SDH::cCANSerial_PEAK](#).

10.27.4.3 virtual tErrorCode SDH::cSerialBase::GetErrorNumber () [inline, virtual]

Helper function that returns the last error number.

- On windows GetLastError() is used
- On cygwin/linux this uses errno

Reimplemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), and [SDH::cTCPSerial](#).

10.27.4.4 char const* SDH::cSerialBase::GetLastErrorMessage (void) [inline]

return the last error message as string. The string returned will be overwritten by the next call to the function

10.27.4.5 virtual double SDH::cSerialBase::GetTimeout () [inline, virtual]

get the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

10.27.4.6 `virtual bool SDH::cSerialBase::IsOpen (void) throw ()` [pure virtual]

Return true if communication channel is open.

Implemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.4.7 `virtual void SDH::cSerialBase::Open (void) throw (cSerialBaseException*)`
[pure virtual]

Open rs232 port *port*.

Open the device with the parameters provided in the constructor

Implemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.4.8 `virtual ssize_t SDH::cSerialBase::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cSerialBaseException*)` [pure virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.4.9 `char * cSerialBase::readline (char * line, int size, char const * eol = "\n", bool return_on_less_data = false) throw (cSerialBaseException*)` [virtual]

Read a line from the device.

A line is terminated with one of the end-of-line (*eol*) characters (

' by default) or until timeout. Up to *size-1* bytes are read and a '\0' char is appended.

Parameters

<i>line</i>	- ptr to where to store the read line
<i>size</i>	- space available in line (bytes)
<i>eol</i>	- a string containing all the chars that mark an end of line
<i>return_on_less_data</i>	- if (<i>return_on_less_data</i> is true (default value), the number of bytes that have been received are returned and the data is stored in <i>data</i> If the <i>return_on_less_data</i> is false, data is only read from serial line, if at least <i>size</i> bytes are available.

A pointer to the line read is returned.

10.27.4.10 `virtual void SDH::cSerialBase::SetTimeout (double timeout) throw (cSerialBaseException*)` [inline, virtual]

set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

Reimplemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.4.11 `virtual bool SDH::cSerialBase::UseCRC16 ()` [inline, virtual]

function that returns true if a CRC16 is used to protect binary communication.

The default is false since only RS232 communication needs this.

Reimplemented in [SDH::cRS232](#), and [SDH::cRS232](#).

10.27.4.12 `virtual int SDH::cSerialBase::write (char const * ptr, int len = 0) throw (cSerialBaseException*)` [pure virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the rs232 device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

Implemented in [SDH::cCANSerial_ESD](#), [SDH::cCANSerial_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), and [SDH::cTCPSerial](#).

10.27.5 Member Data Documentation

10.27.5.1 `cDBG SDH::cSerialBase::dbg`

A stream object to print colored debug messages.

10.27.5.2 `double SDH::cSerialBase::timeout` [protected]

timeout in seconds

10.27.5.3 `char SDH::cSerialBase::ungetch` [protected]

an already read data byte of the next line

10.27.5.4 bool SDH::cSerialBase::ungetch_valid [protected]

Flag, true if ungetch is valid.

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

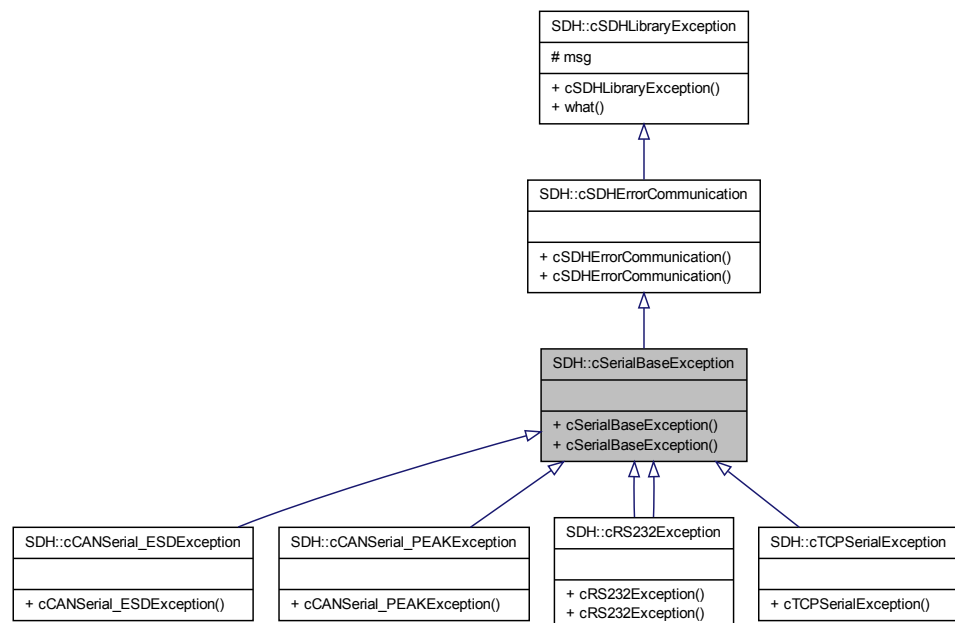
- [sdh/serialbase.h](#)
- [sdh/serialbase.cpp](#)

10.28 SDH::cSerialBaseException Class Reference

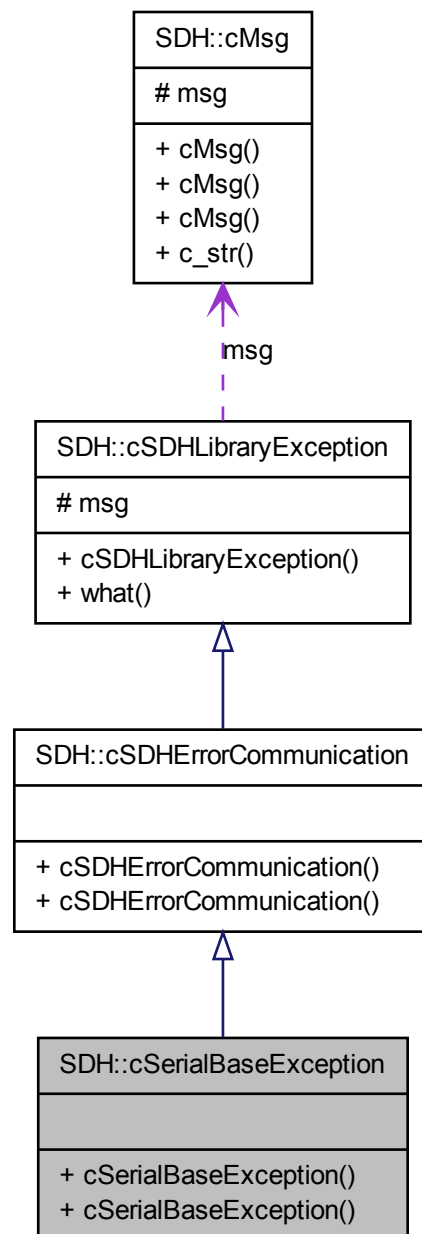
Derived exception class for low-level serial communication related exceptions.

```
#include <serialbase.h>
```

Inheritance diagram for SDH::cSerialBaseException:



Collaboration diagram for SDH::cSerialBaseException:



Public Member Functions

- [cSerialBaseException](#) (cMsg const &_msg)
- [cSerialBaseException](#) (char const *_type, cMsg const &_msg)

10.28.1 Detailed Description

Derived exception class for low-level serial communication related exceptions.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 SDH::cSerialBaseException::cSerialBaseException (cMsg const & *msg*)
[inline]

10.28.2.2 SDH::cSerialBaseException::cSerialBaseException (char const * *_type*, cMsg const & *msg*) [inline]

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

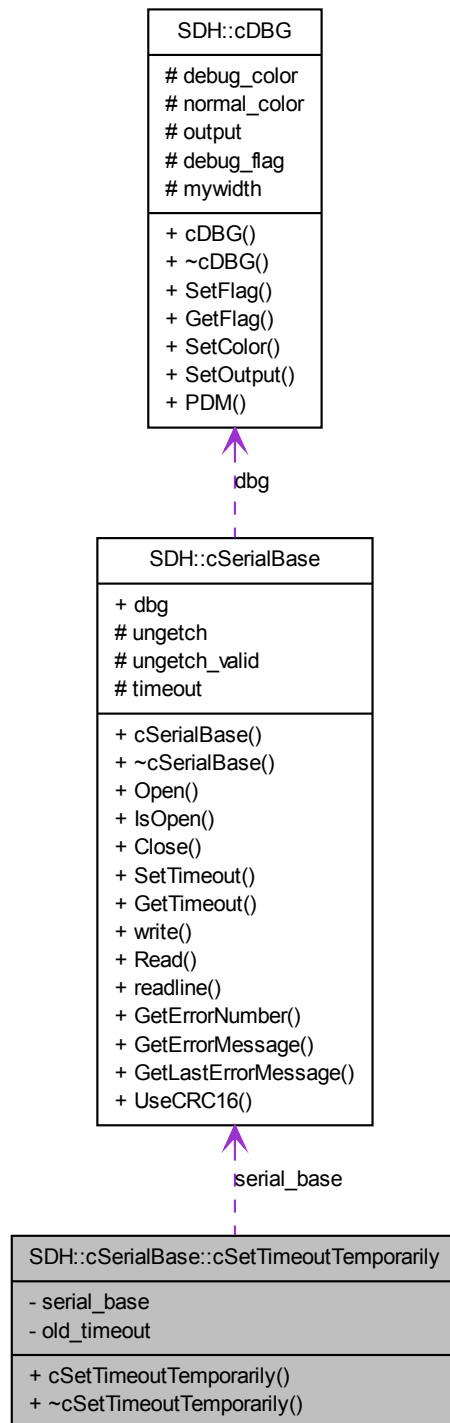
- sdh/[serialbase.h](#)

10.29 SDH::cSerialBase::cSetTimeoutTemporarily Class Reference

helper class to set timeout of _serial_base on construction and reset to previous value on destruction. (RAII-idiom)

```
#include <serialbase.h>
```

Collaboration diagram for SDH::cSerialBase::cSetTimeoutTemporarily:



Public Member Functions

- [cSetTimeoutTemporarily](#) ([cSerialBase](#) *_serial_base, double new_timeout)
CTOR: remember current timeout of _serial_base and set its timeout to new_timeout, but only if current timeout and new_timeout differ.
- [~cSetTimeoutTemporarily](#) ()
DTOR: restore the remembered timeout.

10.29.1 Detailed Description

helper class to set timeout of _serial_base on construction and reset to previous value on destruction. (RAII-idiom)

10.29.2 Constructor & Destructor Documentation

- 10.29.2.1 SDH::cSerialBase::cSetTimeoutTemporarily::cSetTimeoutTemporarily (cSerialBase *_serial_base, double new_timeout) [inline]**

CTOR: remember current timeout of _serial_base and set its timeout to new_timeout, but only if current timeout and new_timeout differ.

- 10.29.2.2 SDH::cSerialBase::cSetTimeoutTemporarily::~~cSetTimeoutTemporarily () [inline]**

DTOR: restore the remembered timeout.

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

- [sdh/serialbase.h](#)

10.30 SDH::cSetValueTemporarily< T > Class Template Reference

helper class to set value on construction and reset to previous value on destruction. (RAII-idiom)

```
#include <util.h>
```

Public Member Functions

- [cSetValueTemporarily](#) (T *_value_ptr, T new_value)
CTOR: remember current value of _value_ptr and set it to new_value.
- [~cSetValueTemporarily](#) ()

DTOR: restore the remembered value.

10.30.1 Detailed Description

`template<typename T> class SDH::cSetValueTemporarily< T >`

helper class to set value on construction and reset to previous value on destruction.
(RAII-idiom)

10.30.2 Constructor & Destructor Documentation

10.30.2.1 `template<typename T> SDH::cSetValueTemporarily< T
>::cSetValueTemporarily (T * _value_ptr, T new_value) [inline]`

CTOR: remember current value of `_value_ptr` and set it to `new_value`.

10.30.2.2 `template<typename T> SDH::cSetValueTemporarily< T
>::~~cSetValueTemporarily () [inline]`

DTOR: restore the remembered value.

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

- [sdh/util.h](#)

10.31 SDH::cSimpleStringList Class Reference

A simple string list. (Fixed maximum number of strings of fixed maximum length)

```
#include <simplestringlist.h>
```

Public Types

- enum { [eMAX_LINES](#) = 256, [eMAX_CHARS](#) = 256 }
anonymous enum instead of define macros

Public Member Functions

- [cSimpleStringList](#) ()
Default constructor: init members.
- char * [CurrentLine](#) ()

Return the current line.

- char * [NextLine](#) ()
Return the next line, this increases current_line.
- int [Length](#) () const
Return number of lines stored.
- char * [operator\[\]](#) (int index)
return ptr to line with index.
- char const * [operator\[\]](#) (int index) const
return ptr to line with index.
- void [Reset](#) ()
reset list

Public Attributes

- int [current_line](#)
the index of the current line. For empty cSimpleStringLists this is -1.

Protected Attributes

- char [line](#) [eMAX_LINES][eMAX_CHARS]
a fixed length array of lines with fixed length

10.31.1 Detailed Description

A simple string list. (Fixed maximum number of strings of fixed maximum length)

10.31.2 Member Enumeration Documentation

10.31.2.1 anonymous enum

anonymous enum instead of define macros

Enumerator:

eMAX_LINES

eMAX_CHARS

10.31.3 Constructor & Destructor Documentation

10.31.3.1 cSimpleStringList::cSimpleStringList ()

Default constructor: init members.

10.31.4 Member Function Documentation

10.31.4.1 char * cSimpleStringList::CurrentLine ()

Return the current line.

10.31.4.2 int cSimpleStringList::Length () const

Return number of lines stored.

10.31.4.3 char * cSimpleStringList::NextLine ()

Return the next line, this increases current_line.

10.31.4.4 char * cSimpleStringList::operator[] (int index)

return ptr to line with index.

if index < 0 then the numbering starts from the end, thus [-1] gives the last line, [-2] the next to last, ...

10.31.4.5 char const * cSimpleStringList::operator[] (int index) const

return ptr to line with index.

if index < 0 then the numbering starts from the end, thus [-1] gives the last line, [-2] the next to last, ...

10.31.4.6 void cSimpleStringList::Reset ()

reset list

10.31.5 Member Data Documentation

10.31.5.1 int SDH::cSimpleStringList::current_line

the index of the current line. For empty cSimpleStringLists this is -1.

10.31.5.2 char SDH::cSimpleStringList::line[eMAX_LINES][eMAX_CHARS] [protected]

a fixed length array of lines with fixed length

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

- [sdh/simplestringlist.h](#)
- [sdh/simplestringlist.cpp](#)

10.32 SDH::cSimpleTime Class Reference

Very simple class to measure elapsed time.

```
#include <simpletime.h>
```

Public Member Functions

- [cSimpleTime](#) ()
Constructor: store current time ("now") internally.
- void [StoreNow](#) (void)
Store current time internally.
- double [Elapsed](#) (void) const
Return time in seconds elapsed between the time stored in the object and now.
- long [Elapsed_us](#) (void) const
Return time in micro seconds elapsed between the time stored in the object and now.
- double [Elapsed](#) (cSimpleTime const &other) const
Return time in seconds elapsed between the time stored in the object and other.
- long [Elapsed_us](#) (cSimpleTime const &other) const
Return time in micro seconds elapsed between the time stored in the object and other.
- timeval [Timeval](#) (void)
Return the time stored as C struct timeval.

Protected Attributes

- struct timeval [a_time](#)

10.32.1 Detailed Description

Very simple class to measure elapsed time.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 SDH::cSimpleTime::cSimpleTime () [inline]

Constructor: store current time ("now") internally.

10.32.3 Member Function Documentation

10.32.3.1 double SDH::cSimpleTime::Elapsed (void) const [inline]

Return time in seconds elapsed between the time stored in the object and now.

10.32.3.2 double SDH::cSimpleTime::Elapsed (cSimpleTime const & *other*) const [inline]

Return time in seconds elapsed between the time stored in the object and *other*.

10.32.3.3 long SDH::cSimpleTime::Elapsed_us (cSimpleTime const & *other*) const [inline]

Return time in micro seconds elapsed between the time stored in the object and *other*.

10.32.3.4 long SDH::cSimpleTime::Elapsed_us (void) const [inline]

Return time in micro seconds elapsed between the time stored in the object and now.

10.32.3.5 void SDH::cSimpleTime::StoreNow (void) [inline]

Store current time internally.

10.32.3.6 timeval SDH::cSimpleTime::Timeval (void) [inline]

Return the time stored as C struct timeval.

10.32.4 Member Data Documentation

10.32.4.1 struct timeval SDH::cSimpleTime::a_time [protected]

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

- [sdh/simpletime.h](#)

10.33 SDH::cSimpleVector Class Reference

A simple vector implementation.

```
#include <simplevector.h>
```

Public Types

- enum { [eNUMBER_OF_ELEMENTS](#) = 7 }
anonymous enum (instead of define like macros)

Public Member Functions

- [cSimpleVector](#) () throw (cSimpleVectorException*)
Default constructor: init members to zero.
- [cSimpleVector](#) (int nb_values, char const *str) throw (cSimpleVectorException*)
Constructor: init members from nb_values comma separated values in the give string str.
- [cSimpleVector](#) (int nb_values, int start_index, char const *str) throw (cSimpleVectorException*)
Constructor: init members from nb_values comma separated values in the give string str.
- [cSimpleVector](#) (int nb_values, int start_index, float *values) throw (cSimpleVectorException*)
Constructor: init members beginning with start_index from nb_values in array values.
- void [FromString](#) (int nb_values, int start_index, char const *str) throw (cSimpleVectorException*)
init nb_values starting from index start_index from comma separated values in str
- double & [operator\[\]](#) (unsigned int index)
index operator, return a reference to the index-th element of this
- double & [x](#) (void)
Interpret object as x/y/z vector: return x = the first element, if that is valid.
- double & [y](#) (void)

Interpret object as x/y/z vector: return x = the first element, if that is valid.

- double & **z** (void)

Interpret object as x/y/z vector: return x = the first element, if that is valid.

- bool **Valid** (unsigned int index) const

Return true if vector element index is valid (has been accessed at least once)

Protected Attributes

- double **value** [eNUMBER_OF_ELEMENTS]
- int **valid**

*bit mask which values in **value** are valid*

10.33.1 Detailed Description

A simple vector implementation. Objects of this class are used to return vector like answers from the SDH firmware to the **cSDHBase** class. End users need not use this class, as **cSDH**, the real end user interface class provides a more convenient way using STL vectors.

10.33.2 Member Enumeration Documentation

10.33.2.1 anonymous enum

anonymous enum (instead of define like macros)

Enumerator:

eNUMBER_OF_ELEMENTS number of elements in vector

10.33.3 Constructor & Destructor Documentation

10.33.3.1 cSimpleVector::cSimpleVector () throw (cSimpleVectorException*)

Default constructor: init members to zero.

10.33.3.2 cSimpleVector::cSimpleVector (int nb_values, char const * str) throw (cSimpleVectorException*)

Constructor: init members from *nb_values* comma separated values in the give string *str*.

10.33.3.3 `cSimpleVector::cSimpleVector (int nb_values, int start_index, char const * str)
throw (cSimpleVectorException*)`

Constructor: init members from *nb_values* comma separated values in the give string *str*.

10.33.3.4 `cSimpleVector::cSimpleVector (int nb_values, int start_index, float * values) throw
(cSimpleVectorException*)`

Constructor: init members beginning with *start_index* from *nb_values* in array *values*.

10.33.4 Member Function Documentation

10.33.4.1 `void cSimpleVector::FromString (int nb_values, int start_index, char const * str)
throw (cSimpleVectorException*)`

init *nb_values* starting from index *start_index* from comma separated values in *str*

10.33.4.2 `double & cSimpleVector::operator[] (unsigned int index)`

index operator, return a reference to the *index-th* element of this

10.33.4.3 `bool cSimpleVector::Valid (unsigned int index) const`

Return true if vector element *index* is valid (has been accessed at least once)

10.33.4.4 `double & cSimpleVector::x (void)`

Interpret object as x/y/z vector: return x = the first element, if that is valid.

10.33.4.5 `double & cSimpleVector::y (void)`

Interpret object as x/y/z vector: return x = the first element, if that is valid.

10.33.4.6 `double & cSimpleVector::z (void)`

Interpret object as x/y/z vector: return x = the first element, if that is valid.

10.33.5 Member Data Documentation

10.33.5.1 `int SDH::cSimpleVector::valid` [protected]

bit mask which values in [value](#) are valid

10.33.5.2 `double SDH::cSimpleVector::value[eNUMBER_OF_ELEMENTS]`
[protected]

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

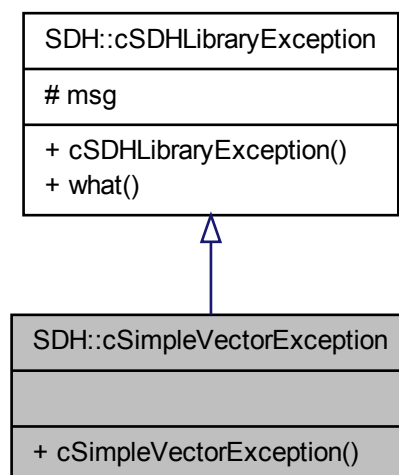
- [sdh/simplevector.h](#)
- [sdh/simplevector.cpp](#)

10.34 SDH::cSimpleVectorException Class Reference

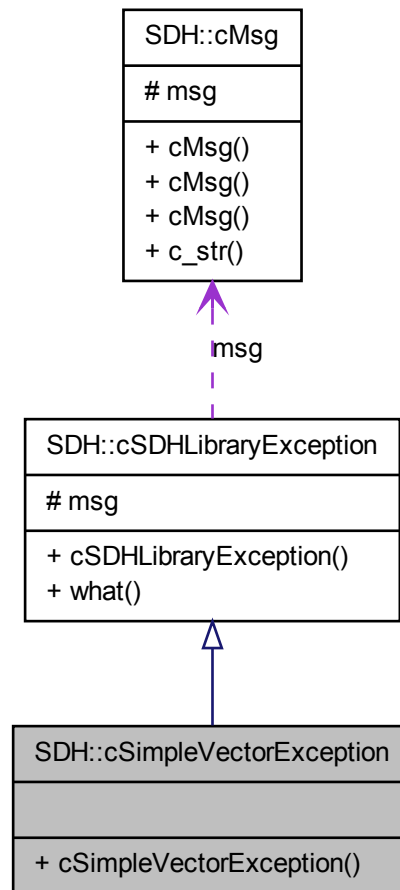
Derived exception class for low-level simple vector related exceptions.

```
#include <simplevector.h>
```

Inheritance diagram for SDH::cSimpleVectorException:



Collaboration diagram for SDH::cSimpleVectorException:



Public Member Functions

- [cSimpleVectorException](#) ([cMsg](#) const &_msg)

10.34.1 Detailed Description

Derived exception class for low-level simple vector related exceptions.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 SDH::cSimpleVectorException::cSimpleVectorException (cMsg const & *_msg*)
[inline]

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

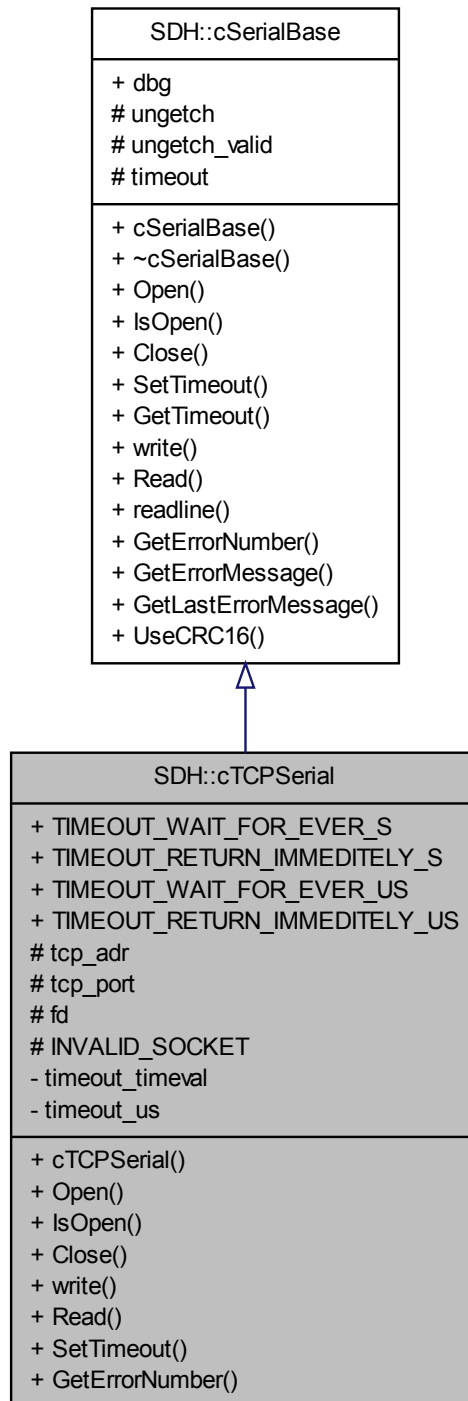
- [sdh/simplevector.h](#)

10.35 SDH::cTCPSerial Class Reference

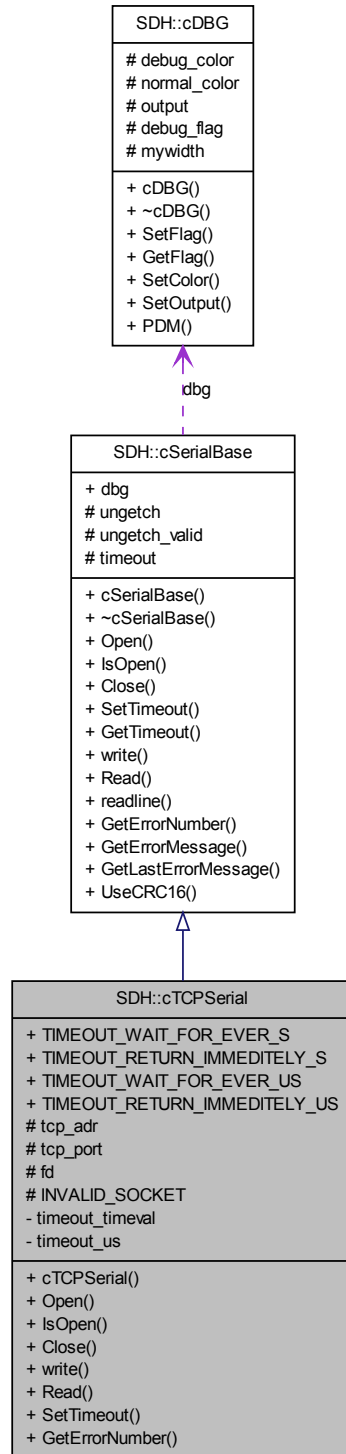
Low-level communication class to access a CAN port.

```
#include <tcpserial.h>
```

Inheritance diagram for SDH::cTCPSerial:



Collaboration diagram for SDH::cTCPSerial:



Public Member Functions

- [cTCPSerial](#) (char const *_tcp_adr, int _tcp_port, double _timeout) throw (cTCPSerialException*)
- void [Open](#) (void) throw (cTCPSerialException*)
- bool [IsOpen](#) (void) throw ()
Return true if interface to CAN ESD is open.
- void [Close](#) (void) throw (cTCPSerialException*)
Close the previously opened CAN ESD interface port.
- int [write](#) (char const *ptr, int len=0) throw (cTCPSerialException*)
Write data to a previously opened port.
- ssize_t [Read](#) (void *data, ssize_t [size](#), long timeout_us, bool return_on_less_data) throw (cTCPSerialException*)
- void [SetTimeout](#) (double _timeout) throw (cSerialBaseException*)
set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)
- virtual [tErrorCode](#) [GetErrorNumber](#) ()

Static Public Attributes

- static double const [TIMEOUT_WAIT_FOR_EVER_S](#) = -1.0
- static double const [TIMEOUT_RETURN_IMMEDIATELY_S](#) = 0.0
- static long const [TIMEOUT_WAIT_FOR_EVER_US](#) = -1
- static long const [TIMEOUT_RETURN_IMMEDIATELY_US](#) = 0

Protected Attributes

- std::string [tcp_adr](#)
- int [tcp_port](#)
the TCP port to use
- int [fd](#)
the file descriptor of the socket

Static Protected Attributes

- static const int [INVALID_SOCKET](#) = -1

10.35.1 Detailed Description

Low-level communication class to access a CAN port.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 cTCPSerial::cTCPSerial (char const * *_tcp_adr*, int *_tcp_port*, double *_timeout*) throw (cTCPSerialException*)

Constructor: constructs an object to communicate with an SDH via TCP on *_tcp_adr* and *_tcp_port*.

Parameters

<i>_tcp_adr</i>	- a string describing the hostname or IP address of the SDH
<i>_tcp_port</i>	- the port number on the SDH
<i>_timeout</i>	- the timeout when receiving / sending data: <ul style="list-style-type: none"> • < 0.0 : no timeout = wait for ever • == 0.0 : zero timeout = return immediately • > 0.0 : timeout in seconds

10.35.3 Member Function Documentation

10.35.3.1 void cTCPSerial::Close (void) throw (cTCPSerialException*) [virtual]

Close the previously opened CAN ESD interface port.

Implements [SDH::cSerialBase](#).

10.35.3.2 virtual tErrorCode SDH::cTCPSerial::GetErrorNumber () [inline, virtual]

Overloaded helper function that returns the last TCP error number.

Reimplemented from [SDH::cSerialBase](#).

10.35.3.3 bool cTCPSerial::IsOpen (void) throw () [virtual]

Return true if interface to CAN ESD is open.

Implements [SDH::cSerialBase](#).

10.35.3.4 void cTCPSerial::Open (void) throw (cTCPSerialException*) [virtual]

Open the device as configured by the parameters given to the constructor

Implements [SDH::cSerialBase](#).

10.35.3.5 `ssize_t cTCPSerial::Read (void * data, ssize_t size, long timeout_us, bool return_on_less_data) throw (cTCPSerialException*)` [virtual]

Read data from device. This function waits until *max_time_us* us passed or the expected number of bytes are received via serial line. if (*return_on_less_data* is true (default value), the number of bytes that have been received are returned and the data is stored in *data* If the *return_on_less_data* is false, data is only read from serial line, if at least *size* bytes are available.

Implements [SDH::cSerialBase](#).

10.35.3.6 `void cTCPSerial::SetTimeout (double timeout) throw (cSerialBaseException*)` [virtual]

set the timeout for next [readline\(\)](#) calls (negative value means: no timeout, wait for ever)

Reimplemented from [SDH::cSerialBase](#).

10.35.3.7 `int cTCPSerial::write (char const * ptr, int len = 0) throw (cTCPSerialException*)` [virtual]

Write data to a previously opened port.

Write *len* bytes from **ptr* to the CAN device

Parameters

<i>ptr</i>	- pointer the byte array to send in memory
<i>len</i>	- number of bytes to send

Returns

the number of bytes actually written

Implements [SDH::cSerialBase](#).

10.35.4 Member Data Documentation

10.35.4.1 `int SDH::cTCPSerial::fd` [protected]

the file descriptor of the socket

10.35.4.2 `const int SDH::cTCPSerial::INVALID_SOCKET = -1` [static, protected]

10.35.4.3 `std::string SDH::cTCPSerial::tcp_adr` [protected]

10.35.4.4 `int SDH::cTCPSerial::tcp_port` [protected]

the TCP port to use

10.35.4.5 `double const cTCPSerial::TIMEOUT_RETURN_IMMEDIATELY_S = 0.0` [static]

10.35.4.6 `long const cTCPSerial::TIMEOUT_RETURN_IMMEDIATELY_US = 0` [static]

10.35.4.7 `double const cTCPSerial::TIMEOUT_WAIT_FOR_EVER_S = -1.0` [static]

10.35.4.8 `long const cTCPSerial::TIMEOUT_WAIT_FOR_EVER_US = -1` [static]

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

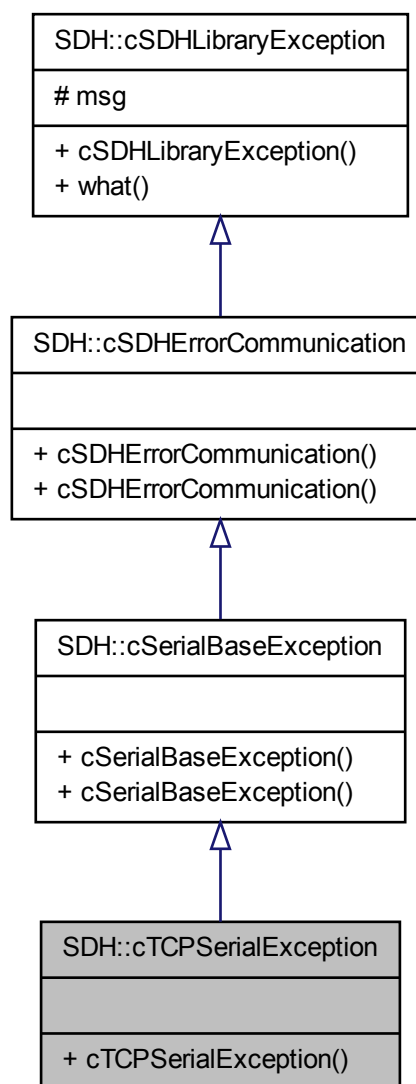
- [sdh/tcpserial.h](#)
- [sdh/tcpserial.cpp](#)

10.36 SDH::cTCPSerialException Class Reference

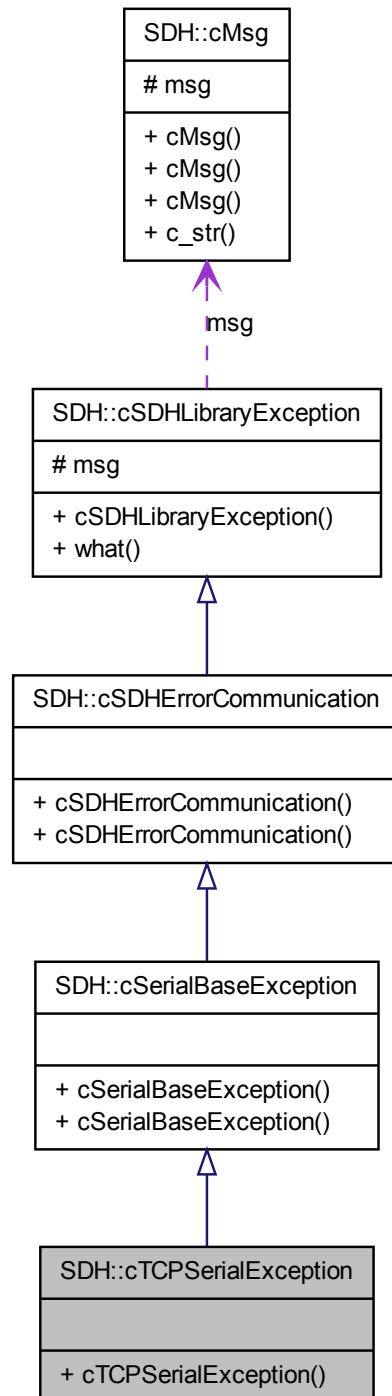
Derived exception class for low-level CAN ESD related exceptions.

```
#include <tcpserial.h>
```

Inheritance diagram for SDH::cTCPSerialException:



Collaboration diagram for SDH::cTCPSerialException:



Public Member Functions

- [cTCPSerialException](#) (cMsg const &_msg)

10.36.1 Detailed Description

Derived exception class for low-level CAN ESD related exceptions.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 SDH::cTCPSerialException::cTCPSerialException (cMsg const & _msg)
[inline]

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

- [sdh/tcpserial.h](#)

10.37 SDH::cUnitConverter Class Reference

Unit conversion class to convert values between physical unit systems.

```
#include <unit_converter.h>
```

Public Member Functions

- [cUnitConverter](#) (char const *_kind, char const *_name, char const *_symbol, double _factor=1.0, double _offset=0.0, int _decimal_places=1)
- double [ToExternal](#) (double internal) const
- [cSimpleVector ToExternal](#) (cSimpleVector &internal) const
- std::vector< double > [ToExternal](#) (std::vector< double > const &internal) const
- double [ToInternal](#) (double external) const
- [cSimpleVector ToInternal](#) (cSimpleVector &external) const
- std::vector< double > [ToInternal](#) (std::vector< double > const &external) const
- char const * [GetKind](#) (void) const
Return the kind of unit converted (something like "angle" or "time")
- char const * [GetName](#) (void) const
Return the name of the external unit (something like "degrees" or "milliseconds")
- char const * [GetSymbol](#) (void) const
Return the symbol of the external unit (something like "deg" or "ms")
- double [GetFactor](#) (void) const
Return the conversion factor from internal to external units.

- double [GetOffset](#) (void) const
Return the conversion offset from internal to external units.
- int [GetDecimalPlaces](#) (void) const
Return the number of decimal places for printing values in the external unit system.

Protected Attributes

- char const * [kind](#)
the kind of unit to be converted (something like "angle" or "time")
- char const * [name](#)
the name of the external unit (something like "degrees" or "milliseconds")
- char const * [symbol](#)
the symbol of the external unit (something like "deg" or "ms")
- double [factor](#)
the conversion factor from internal to external units
- double [offset](#)
the conversion offset from internal to external units
- int [decimal_places](#)
A usefull number of decimal places for printing values in the external unit system.

10.37.1 Detailed Description

Unit conversion class to convert values between physical unit systems. An object of this class can be configured to convert values of a physical unit between 2 physical unit systems. An angle value given in degrees can e.g. be converted from/to radians or vice versa by an object of this class.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 `cUnitConverter::cUnitConverter (char const * _kind, char const * _name, char const * _symbol, double _factor = 1 . 0, double _offset = 0 . 0, int _decimal_places = 1)`

Constructor of [cUnitConverter](#) class.

At construction time the conversion parameters - a *factor* and an *offset* - must be provided along with elements that describe the unit of a value

Parameters

<i>_kind</i>	- a string describing the kind of unit to be converted (something like "angle" or "time")
<i>_name</i>	- the name of the external unit (something like "degrees" or "milliseconds")
<i>_symbol</i>	- the symbol of the external unit (something like "deg" or "ms")
<i>_factor</i>	- the conversion factor from internal to external units
<i>_offset</i>	- the conversion offset from internal to external units
<i>_decimal_places</i>	- A usefull number of decimal places for printing values in the external unit system

Attention

The strings given *_kind*, *_name* and *_symbol* are **NOT** copied, just their address is stored

10.37.3 Member Function Documentation**10.37.3.1 int cUnitConverter::GetDecimalPlaces (void) const**

Return the number of decimal places for printing values in the external unit system.

10.37.3.2 double cUnitConverter::GetFactor (void) const

Return the conversion factor from internal to external units.

10.37.3.3 char const * cUnitConverter::GetKind (void) const

Return the kind of unit converted (something like "angle" or "time")

10.37.3.4 char const * cUnitConverter::GetName (void) const

Return the name of the external unit (something like "degrees" or "milliseconds")

10.37.3.5 double cUnitConverter::GetOffset (void) const

Return the conversion offset from internal to external units.

10.37.3.6 char const * cUnitConverter::GetSymbol (void) const

Return the symbol of the external unit (something like "deg" or "ms")

10.37.3.7 `std::vector< double > cUnitConverter::ToExternal (std::vector< double > const & internal) const`

Convert values in vector *internal*, each given in internal units into external units. Returns a vector with each element converted with $internal[i] * factor + offset$

10.37.3.8 `cSimpleVector cUnitConverter::ToExternal (cSimpleVector & internal) const`

Convert values in simple array *internal*, each given in internal units into external units. Returns a simple array with each valid element converted with $internal[i] * factor + offset$

Only valid entries of the *internal* vector are converted.

10.37.3.9 `double cUnitConverter::ToExternal (double internal) const`

Convert single value *internal* given in internal units into external units. Returns $internal * factor + offset$

10.37.3.10 `double cUnitConverter::ToInternal (double external) const`

Convert value *external* given in external units into internal units. Returns $(external - offset) / factor$

10.37.3.11 `cSimpleVector cUnitConverter::ToInternal (cSimpleVector & external) const`

Convert values in simple array *external*, each given in external units into internal units. Returns a simple array with each valid element converted with $external[i] * factor + offset$

Only valid entries of the *external* vector are converted.

10.37.3.12 `std::vector< double > cUnitConverter::ToInternal (std::vector< double > const & external) const`

Convert values in vector *external*, each given in external units into internal units. Returns a vector with each valid element converted with $external[i] * factor + offset$

10.37.4 Member Data Documentation

10.37.4.1 `int SDH::cUnitConverter::decimal_places` [protected]

A usefull number of decimal places for printing values in the external unit system.

10.37.4.2 `double SDH::cUnitConverter::factor` [protected]

the conversion factor from internal to external units

10.37.4.3 `char const* SDH::cUnitConverter::kind` [protected]

the kind of unit to be converted (something like "angle" or "time")

10.37.4.4 `char const* SDH::cUnitConverter::name` [protected]

the name of the external unit (something like "degrees" or "milliseconds")

10.37.4.5 `double SDH::cUnitConverter::offset` [protected]

the conversion offset from internal to external units

10.37.4.6 `char const* SDH::cUnitConverter::symbol` [protected]

the symbol of the external unit (something like "deg" or "ms")

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

- [sdh/unit_converter.h](#)
- [sdh/unit_converter.cpp](#)

10.38 option Struct Reference

```
#include <getopt.h>
```

Public Attributes

- `char * name`
- `int has_arg`
- `int * flag`
- `int val`

10.38.1 Member Data Documentation

10.38.1.1 `int* option::flag`

10.38.1.2 `int option::has_arg`

10.38.1.3 `char* option::name`

10.38.1.4 `int option::val`

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

- [vcc/getopt.h](#)

10.39 SDH::cDSA::sContactInfo Struct Reference

Structure to hold info about the contact of one sensor patch.

```
#include <dsa.h>
```

Public Attributes

- double [force](#)
accumulated force in N
- double [area](#)
*area of contact in mm*mm.*
- double [cog_x](#)
center of gravity of contact in x direction of sensor patch in mm
- double [cog_y](#)
center of gravity of contact in y direction of sensor patch in mm

10.39.1 Detailed Description

Structure to hold info about the contact of one sensor patch.

10.39.2 Member Data Documentation

10.39.2.1 `double SDH::cDSA::sContactInfo::area`

area of contact in mm*mm.

10.39.2.2 double SDH::cDSA::sContactInfo::cog_x

center of gravity of contact in x direction of sensor patch in mm

10.39.2.3 double SDH::cDSA::sContactInfo::cog_y

center of gravity of contact in y direction of sensor patch in mm

10.39.2.4 double SDH::cDSA::sContactInfo::force

accumulated force in N

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

- [sdh/dsa.h](#)

10.40 SDH::cDSA::sControllerInfo Struct Reference

A data structure describing the controller info about the remote DSACON32m controller.

```
#include <dsa.h>
```

Public Attributes

- [UInt16 error_code](#)
- [UInt32 serial_no](#)
- [UInt8 hw_version](#)
- [UInt16 sw_version](#)
- [UInt8 status_flags](#)
- [UInt8 feature_flags](#)
- [UInt8 senscon_type](#)
- [UInt8 active_interface](#)
- [UInt32 can_baudrate](#)
- [UInt16 can_id](#)

10.40.1 Detailed Description

A data structure describing the controller info about the remote DSACON32m controller.

10.40.2 Member Data Documentation

10.40.2.1 UInt8 SDH::cDSA::sControllerInfo::active_interface

10.40.2.2 UInt32 SDH::cDSA::sControllerInfo::can_baudrate

10.40.2.3 UInt16 SDH::cDSA::sControllerInfo::can_id

10.40.2.4 UInt16 SDH::cDSA::sControllerInfo::error_code

10.40.2.5 UInt8 SDH::cDSA::sControllerInfo::feature_flags

10.40.2.6 UInt8 SDH::cDSA::sControllerInfo::hw_version

10.40.2.7 UInt8 SDH::cDSA::sControllerInfo::senscon_type

10.40.2.8 UInt32 SDH::cDSA::sControllerInfo::serial_no

10.40.2.9 UInt8 SDH::cDSA::sControllerInfo::status_flags

10.40.2.10 UInt16 SDH::cDSA::sControllerInfo::sw_version

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

- [sdh/dsa.h](#)

10.41 SDH::cDSA::sMatrixInfo Struct Reference

A data structure describing a single sensor matrix connected to the remote DSA CON32m controller.

```
#include <dsa.h>
```

Public Attributes

- [UInt16 error_code](#)
- float [texel_width](#)
- float [texel_height](#)
- [UInt16 cells_x](#)
- [UInt16 cells_y](#)
- [UInt8 uid](#) [6]
- [UInt8 reserved](#) [2]
- [UInt8 hw_revision](#)
- float [matrix_center_x](#)
- float [matrix_center_y](#)
- float [matrix_center_z](#)
- float [matrix_theta_x](#)

- float [matrix_theta_y](#)
- float [matrix_theta_z](#)
- float [fullscale](#)
- UInt8 [feature_flags](#)

10.41.1 Detailed Description

A data structure describing a single sensor matrix connected to the remote DSACON32m controller.

10.41.2 Member Data Documentation

10.41.2.1 UInt16 SDH::cDSA::sMatrixInfo::cells_x

10.41.2.2 UInt16 SDH::cDSA::sMatrixInfo::cells_y

10.41.2.3 UInt16 SDH::cDSA::sMatrixInfo::error_code

10.41.2.4 UInt8 SDH::cDSA::sMatrixInfo::feature_flags

10.41.2.5 float SDH::cDSA::sMatrixInfo::fullscale

10.41.2.6 UInt8 SDH::cDSA::sMatrixInfo::hw_revision

10.41.2.7 float SDH::cDSA::sMatrixInfo::matrix_center_x

10.41.2.8 float SDH::cDSA::sMatrixInfo::matrix_center_y

10.41.2.9 float SDH::cDSA::sMatrixInfo::matrix_center_z

10.41.2.10 float SDH::cDSA::sMatrixInfo::matrix_theta_x

10.41.2.11 float SDH::cDSA::sMatrixInfo::matrix_theta_y

10.41.2.12 float SDH::cDSA::sMatrixInfo::matrix_theta_z

10.41.2.13 UInt8 SDH::cDSA::sMatrixInfo::reserved[2]

10.41.2.14 float SDH::cDSA::sMatrixInfo::texel_height

10.41.2.15 float SDH::cDSA::sMatrixInfo::texel_width

10.41.2.16 UInt8 SDH::cDSA::sMatrixInfo::uid[6]

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

- [sdh/dsa.h](#)

10.42 sRecordedData Struct Reference

structure to hold current hand state while recording with demo-benchmark

Public Member Functions

- [sRecordedData](#) (double *_t*, std::vector< double > const &*_aaa*, std::vector< double > const &*_aav*, std::vector< double > const &*_atv*)

Public Attributes

- double *t*
- std::vector< double > *aaa*
- std::vector< double > *aav*
- std::vector< double > *atv*

10.42.1 Detailed Description

structure to hold current hand state while recording with demo-benchmark

10.42.2 Constructor & Destructor Documentation

10.42.2.1 `sRecordedData::sRecordedData (double _t, std::vector< double > const & _aaa, std::vector< double > const & _aav, std::vector< double > const & _atv)`
`[inline]`

10.42.3 Member Data Documentation

10.42.3.1 `std::vector<double> sRecordedData::aaa`

10.42.3.2 `std::vector<double> sRecordedData::aav`

10.42.3.3 `std::vector<double> sRecordedData::atv`

10.42.3.4 `double sRecordedData::t`

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

- demo/[demo-benchmark.cpp](#)

10.43 SDH::cDSA::sResponse Struct Reference

data structure for storing responses from the remote DSACon32m controller

```
#include <dsa.h>
```

Public Member Functions

- [sResponse](#) ([UInt8](#) *_payload, int _max_payload_size)
constructor to init pointer and max size

Public Attributes

- [UInt8](#) packet_id
- [UInt16](#) size
- [UInt8](#) * payload
- [Int32](#) max_payload_size

10.43.1 Detailed Description

data structure for storing responses from the remote DSACON32m controller

10.43.2 Constructor & Destructor Documentation

10.43.2.1 [SDH::cDSA::sResponse::sResponse](#) ([UInt8](#) * _payload, int _max_payload_size)
[inline]

constructor to init pointer and max size

10.43.3 Member Data Documentation

10.43.3.1 [Int32](#) [SDH::cDSA::sResponse::max_payload_size](#)

10.43.3.2 [UInt8](#) [SDH::cDSA::sResponse::packet_id](#)

10.43.3.3 [UInt8*](#) [SDH::cDSA::sResponse::payload](#)

10.43.3.4 [UInt16](#) [SDH::cDSA::sResponse::size](#)

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

- [sdh/dsa.h](#)

10.44 SDH::sSDHBinaryRequest Struct Reference

data structure with binary data for request from PC to [SDH](#)

Public Member Functions

- [sSDHBinaryRequest](#) ([eCommandCode](#) command, double *value, bool use_crc16)
- [tCRCValue * CRC16](#) () const
return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)
- int [GetNbBytesToSend](#) () const
return the total number of bytes to send

Public Attributes

- unsigned char [cmd_code](#)
- unsigned char [nb_data_bytes](#)
- unsigned char [nb_valid_parameters](#)
- union {
 float [parameter](#) [[eNUMBER_OF_ELEMENTS](#)]
 unsigned char [parameter_bytes](#) [sizeof(float)*[eNUMBER_OF_ELEMENTS](#)+sizeof([tCRCValue](#))]
 } [__attribute__](#)

10.44.1 Detailed Description

data structure with binary data for request from PC to [SDH](#)

10.44.2 Constructor & Destructor Documentation

10.44.2.1 [sSDHBinaryRequest::sSDHBinaryRequest](#) ([eCommandCode](#) *command*, double **value*, bool *use_crc16*)

ctor, create a request with cmd_code *command* and [eNUMBER_OF_ELEMENTS](#) parameter from *value* or no parameters if *value* is NULL. Add crc if *use_crc16* is true and set nb_data_bytes appropriately

10.44.3 Member Function Documentation

10.44.3.1 [tCRCValue* SDH::sSDHBinaryRequest::CRC16](#) () const [inline]

return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)

10.44.3.2 [int SDH::sSDHBinaryRequest::GetNbBytesToSend](#) () const [inline]

return the total number of bytes to send

10.44.4 Member Data Documentation

- 10.44.4.1 union { ... } SDH::sSDHBinaryRequest::__attribute__
- 10.44.4.2 unsigned char SDH::sSDHBinaryRequest::cmd_code
- 10.44.4.3 unsigned char SDH::sSDHBinaryRequest::nb_data_bytes
- 10.44.4.4 unsigned char SDH::sSDHBinaryRequest::nb_valid_parameters
- 10.44.4.5 float SDH::sSDHBinaryRequest::parameter[eNUMBER_OF_ELEMENTS]
- 10.44.4.6 unsigned char SDH::sSDHBinaryRequest::parameter_bytes[sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]

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

- [sdh/sdhserial.cpp](#)

10.45 SDH::sSDHBinaryResponse Struct Reference

data structure with binary data for response from [SDH](#) to PC

Public Member Functions

- [tCRCValue](#) * [CRC16](#) () const
return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)
- void [CheckCRC16](#) () const throw (cSDHErrorCommunication*)
check the CRC value in parameter_bytes. Throw an exception if check fails

Public Attributes

- unsigned char [cmd_code](#)
- unsigned char [nb_data_bytes](#)
- unsigned char [nb_valid_parameters](#)
- unsigned char [status_code](#)
- union {
 float [parameter](#) [eNUMBER_OF_ELEMENTS]
 unsigned char [parameter_bytes](#) [sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]
};

10.45.1 Detailed Description

data structure with binary data for response from [SDH](#) to PC

10.45.2 Member Function Documentation

10.45.2.1 void `sSDHBinaryResponse::CheckCRC16 () const` throw
(`cSDHErrorCommunication*`)

check the CRC value in parameter_bytes. Throw an exception if check fails

10.45.2.2 `tCRCValue*` `SDH::sSDHBinaryResponse::CRC16 () const` [inline]

return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)

10.45.3 Member Data Documentation

10.45.3.1 union { ... }

10.45.3.2 unsigned char `SDH::sSDHBinaryResponse::cmd_code`

10.45.3.3 unsigned char `SDH::sSDHBinaryResponse::nb_data_bytes`

10.45.3.4 unsigned char `SDH::sSDHBinaryResponse::nb_valid_parameters`

10.45.3.5 float `SDH::sSDHBinaryResponse::parameter[eNUMBER_OF_ELEMENTS]`

10.45.3.6 unsigned char `SDH::sSDHBinaryResponse::parameter_ - bytes[sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]`

10.45.3.7 unsigned char `SDH::sSDHBinaryResponse::status_code`

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

- [sdh/sdhserial.cpp](#)

10.46 SDH::cDSA::sSensitivityInfo Struct Reference

Structure to hold info about the sensitivity settings of one sensor patch.

```
#include <dsa.h>
```

Public Attributes

- [UInt16 error_code](#)
0000h, if successful, otherwise error code
- [UInt8 adj_flags](#)
- float [cur_sens](#)
- float [fact_sens](#)

10.46.1 Detailed Description

Structure to hold info about the sensitivity settings of one sensor patch.

10.46.2 Member Data Documentation

10.46.2.1 [UInt8 SDH::cDSA::sSensitivityInfo::adj_flags](#)

Bit vector indicating the sensitivity adjustment options:

- D7...2 reserved
- D1 1: user can change the sensitivity 0: sensitivity cannot be changed by the user

10.46.2.2 [float SDH::cDSA::sSensitivityInfo::cur_sens](#)

Currently set sensitivity value. Floating point value. 0 is minimum sensitivity, 1.0 is maximum sensitivity.

10.46.2.3 [UInt16 SDH::cDSA::sSensitivityInfo::error_code](#)

0000h, if successful, otherwise error code

10.46.2.4 [float SDH::cDSA::sSensitivityInfo::fact_sens](#)

Sensitivity value that is used, if a factory restore command is issued. Floating point value. 0 is minimum sensitivity, 1.0 is maximum sensitivity.

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

- [sdh/dsa.h](#)

10.47 SDH::cDSA::sSensorInfo Struct Reference

A data structure describing the sensor info about the remote DSACON32m controller.

```
#include <dsa.h>
```

Public Attributes

- [UInt16 error_code](#)
- [UInt16 nb_matrices](#)
- [UInt16 generated_by](#)
- [UInt8 hw_revision](#)
- [UInt32 serial_no](#)
- [UInt8 feature_flags](#)

10.47.1 Detailed Description

A data structure describing the sensor info about the remote DSACON32m controller.

10.47.2 Member Data Documentation

10.47.2.1 [UInt16 SDH::cDSA::sSensorInfo::error_code](#)

10.47.2.2 [UInt8 SDH::cDSA::sSensorInfo::feature_flags](#)

10.47.2.3 [UInt16 SDH::cDSA::sSensorInfo::generated_by](#)

10.47.2.4 [UInt8 SDH::cDSA::sSensorInfo::hw_revision](#)

10.47.2.5 [UInt16 SDH::cDSA::sSensorInfo::nb_matrices](#)

10.47.2.6 [UInt32 SDH::cDSA::sSensorInfo::serial_no](#)

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

- [sdh/dsa.h](#)

10.48 SDH::cDSA::sTactileSensorFrame Struct Reference

A data structure describing a full tactile sensor frame read from the remote DSACON32m controller.

```
#include <dsa.h>
```

Public Member Functions

- [sTactileSensorFrame](#) (void)

constructor

Public Attributes

- [UInt32](#) timestamp

the timestamp of the frame. Use [GetAgeOfFrame\(\)](#) to set this into relation with the time of the PC.

- [UInt8](#) flags

internal data

- [tTexel](#) * texel

an 2D array of tTexel elements. Use [GetTexel\(\)](#) for easy access to specific individual elements.

10.48.1 Detailed Description

A data structure describing a full tactile sensor frame read from the remote DSACON32m controller.

Remarks

- An object of this type is stored within the [cDSA](#) object
- You can get a reference to the [sTactileSensorFrame](#) object with the [GetFrame\(\)](#) function.
- The object must be updated manually (for now)
 - by setting an appropriate framerate with [SetFramerate\(\)](#) once
 - by calling [UpdateFrame\(\)](#) periodically
-

10.48.2 Constructor & Destructor Documentation

10.48.2.1 `SDH::cDSA::sTactileSensorFrame::sTactileSensorFrame (void)` `[inline]`

constructor

10.48.3 Member Data Documentation

10.48.3.1 `UInt8 SDH::cDSA::sTactileSensorFrame::flags`

internal data

10.48.3.2 tTexel* SDH::cDSA::sTactileSensorFrame::texel

an 2D array of tTexel elements. Use [GetTexel\(\)](#) for easy access to specific individual elements.

10.48.3.3 UInt32 SDH::cDSA::sTactileSensorFrame::timestamp

the timestamp of the frame. Use [GetAgeOfFrame\(\)](#) to set this into relation with the time of the PC.

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

- [sdh/dsa.h](#)

Chapter 11

File Documentation

11.1 `architecture.dox` File Reference

Short overview of the SDHLibrary-CPP and [SDH](#) architecture.

11.1.1 Detailed Description

Short overview of the SDHLibrary-CPP and [SDH](#) architecture.

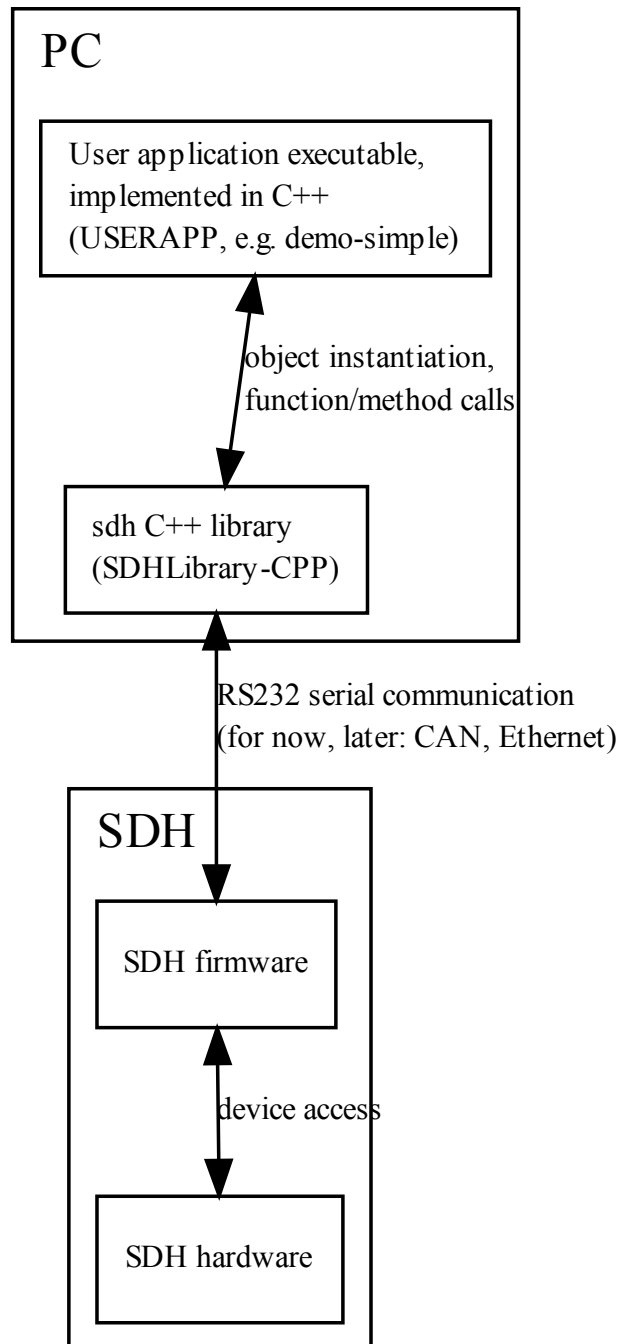
11.1.2 Overview

Naming convention:

As a convention "**SDH**" (capital letters) is used to refer to the physical device, the three fingered SCHUNK Dexterous Hand, while "**sdh**" (small letters) refers to the PC-software that communicates with the physical [SDH](#) device. Within the "sdh" PC-software further entities can be distinguished: The C++ library **SDHLibrary-CPP.so** (on Linux) or **SDHLibrary-CPP.dll** (on Windows/cygwin) that contains the complete sdh library including the user interface class [SDH::cSDH](#). This [SDH::cSDH](#) class will be described in detail below.

Basic structure:

The basic structure of the components looks like this:



Basic architecture:

There are several classes defined in SDHLibrary-CPP:

- [SDH::cSDH](#) the class used to communicate with the [SDH](#). This class provides the functional interface of the [SDH](#). It should be used by end users, as its interface is considered more stable.
- Other classes, like [cSDHBase](#) and [cSDHSerial](#), are used by [SDH::cSDH](#) and provide more low level services and should **NOT** be used directly, as their interfaces are subject to change in future releases.
- [cSDHLibraryException](#) and derivatives: these are used when an exception is raised

Example use:

An exemplary use of the `sdh` module in a user application in C++ might look like this:

```
...
// Include the cSDH interface
include <sdh.h>

// Create an instance "hand" of the class cSDH:
cSDH hand;

// Open communication to the SDH device via default serial port 0 == "COM
1"
hand.OpenRS232();

// Perform some action:
//   get the current actual axis angles of finger 0
std::vector<double> faa = hand.GetFingerActualAngle( 0 );

//   modify these by decreasing the proximal and the distal axis angles:
std::vector<double> fta = faa;
fta[1] -= 10;
fta[2] -= 10;

//   set modified angles as new target angles:
hand.SetFingerTargetAngle( 0, fta );

//   now make the finger move there:
hand.MoveFinger( 0 );

// Finally close connection to SDH again (This automatically
// switches off the axis controllers to prevent overheating):
hand.Close();
```

Real example code is available in the `demo-*.cpp` code files, see e.g.

- [demo-simple.cpp](#)
- [demo-temperature.cpp](#)
- [demo-GetAxisActualAngle.cpp](#)

11.2 connectors.dox File Reference

11.2.1 Detailed Description

11.2.2 General project information

Author

Dirk Osswald

Date

2007-02-19

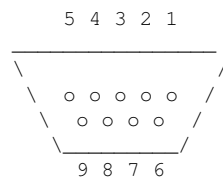
11.2.3 Purpose

This page describes the connectors of the [SDH](#) / LWA

SDH

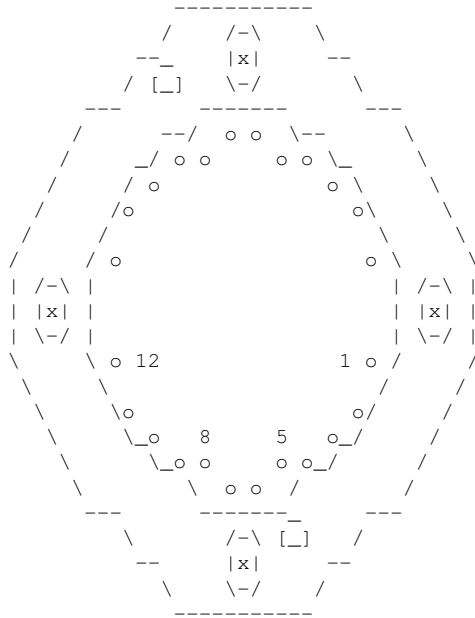
RS232 Communication connector cable of the [SDH](#) at the base of the LWA:

- 9pin Sub-D female connector (View from connecting side):

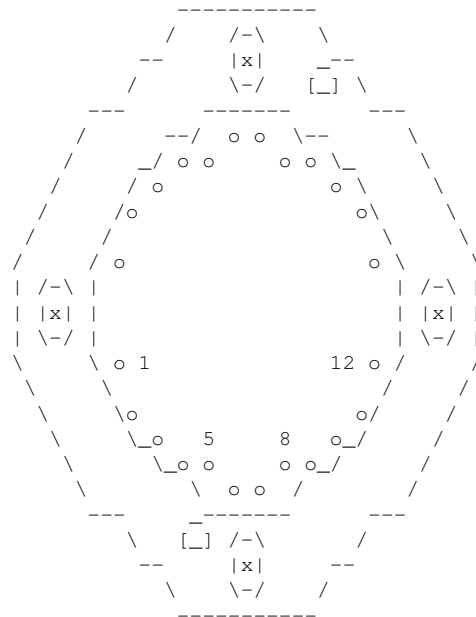


- 1 : here: nc, (normally: Data Carrier Detect)
 - 2 : RxD of PC = TxD of [SDH](#)
 - 3 : TxD of PC = RxD of [SDH](#)
 - 4 : here: nc, (normally: Data Terminal Ready)
 - 5 : GND
 - 6 : here: nc, (normally: Data Set Ready)
 - 7 : here: nc, (normally: Request To Send)
 - 8 : here: nc, (normally: Clear To Send)
 - 9 : here: nc, (Ring Indicator)
-
- RS232 + Power Communication connector:

- 2x12pin FWS connector, view from connecting side of FWS part mounted on LWA: only one half is used for the [SDH](#)



- 1 : Pwr (RD) 24V supply for power and logic of SDH2
 - 2 : TxD (BN) RS232 TxD of PC = RxD of SDH2
 - 3 : RxD (WH) RS232 RxD of PC = TxD of SDH2
 - 4 : TxD2 (RD) RS232 TxD of PC = RxD of SDH2, second RS232 channel of SDH2
 - 5 : RxD2 (PK) RS232 RxD of PC = TxD of SDH2, second RS232 channel of SDH2
 - 6 : CAN_H (GN) CAN High
 - 7 : CAN_L (YE) CAN Low
 - 8 : ENET TPI+ Ethernet
 - 9 : ENET TPI- Ethernet
 - 10 : ENET TPO+ Ethernet
 - 11 : ENET TPO- Ethernet
 - 12 : GND (BK) Ground
- 2x12pin FWS connector view from connecting side of FWS part mounted on [SDH](#): only one half is used by the [SDH](#)

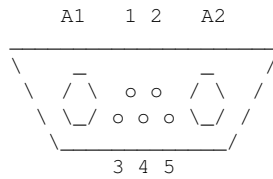


- 1 : Pwr (RD) 24V supply for power and logic of SDH2
- 2 : TxD (BN) RS232 Tx of PC = Rx of SDH2
- 3 : Rx (WH) RS232 Rx of PC = Tx of SDH2
- 4 : Tx2 (RD) RS232 Tx of PC = Rx of SDH2, second RS232 channel of SDH2
- 5 : Rx2 (PK) RS232 Rx of PC = Tx of SDH2, second RS232 channel of SDH2
- 6 : CAN_H (GN) CAN High
- 7 : CAN_L (YE) CAN Low
- 8 : ENET TPI+ Ethernet
- 9 : ENET TPI- Ethernet
- 10 : ENET TPO+ Ethernet
- 11 : ENET TPO- Ethernet
- 12 : GND (BK) Ground

LWA

Combined power / CAN connector of the LWA:

- Male connector (View from connecting side):

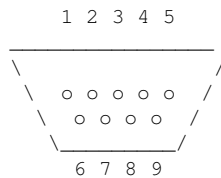


- A1 : +24V
- A2 : 0V (blue)
- 1 : CAN-High (Profibus-A1) (green)
- 2 : RxD
- 3 : CAN-Low (profibus-B1) (yellow)
- 4 : Shield
- 5 : TxD

ESD CAN-USB mini

On the PC-side CAN interface cards are often connected as follows:

- 9pin Sub-D Male connector (View from connecting side):



- 1 : reserved
- 2 : CAN_L (yellow)
- 3 : CAN_GND (blue)
- 4 : reserved
- 5 : Shield
- 6 : (CAN_GND)
- 7 : CAN_H (green)
- 8 : reserved
- 9 : reserved

- char const * [__version__](#) = "\$Id: cancat.cpp 9739 2013-02-04 15:39:29Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.3.1 Detailed Description

Yet incomplete tool to send and receive data via CAN. See [__help__](#) and online help ("-h" or "--help") for available options.

11.3.2 General file information

Author

Dirk Osswald

Date

2007-01-18

11.3.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.3.4 Function Documentation

11.3.4.1 int main (int *argc*, char ** *argv*)

11.3.5 Variable Documentation

11.3.5.1 char const* [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"

11.3.5.2 char const* [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.3.5.3 char const* [__help__](#) = "Send data from command line via ESD or PEAK CAN and display replies until CTRL-C is pressed."

11.3.5.4 char const* [__url__](#) = "http://www.schunk.com"

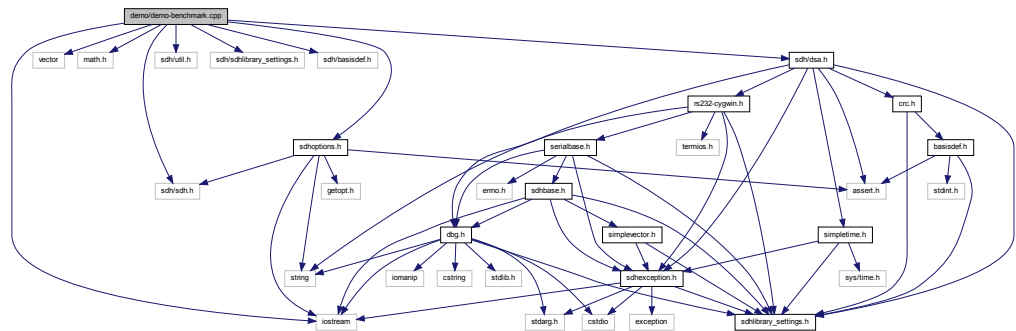
11.3.5.5 char const* [__version__](#) = "\$Id: cancat.cpp 9739 2013-02-04 15:39:29Z Osswald2 \$"

11.4 demo/demo-benchmark.cpp File Reference

Simple script to benchmark communication speed of the [SDH](#). See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
```

Include dependency graph for demo-benchmark.cpp:



- struct `sRecordedData`
structure to hold current hand state while recording with demo-benchmark

- #define _USE_MATH_DEFINES
- #define DEMO_BENCHMARK_USE_COMBINED_SET_GET 1

- void **GotoPose** (cSDH &hand, std::vector< double > &ta)
- int **main** (int argc, char **argv)

- `char const * usage = "usage: demo-benchmark [options]\n"`

- `cDBG cdbg` (false,"red")

Some informative variables

- `char const * __help__` = " > demo-benchmark --port=2 --dsaport=3 -v\n"
- `char const * __author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- `char const * __url__` = "http://www.schunk.com"
- `char const * __version__` = "\$Id: demo-benchmark.cpp 10895 2013-11-11 14:15:08Z Osswald2 \$"
- `char const * __copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.4.1 Detailed Description

Simple script to benchmark communication speed of the [SDH](#). See `__help__` and online help ("-h" or "--help") for available options.

11.4.2 General file information

Author

Dirk Osswald

Date

2011-01-31

11.4.3 Copyright

Copyright (c) 2011 SCHUNK GmbH & Co. KG

11.4.4 Define Documentation

11.4.4.1 `#define _USE_MATH_DEFINES`

11.4.4.2 `#define DEMO_BENCHMARK_USE_COMBINED_SET_GET 1`

11.4.5 Function Documentation

11.4.5.1 `void GotoPose (cSDH & hand, std::vector< double > & ta)`

Move all axes of *hand* to position *ta* using the coordinated pose controller

Parameters

<i>hand</i>	- reference to the SDH to operate on
<i>ta</i>	- axis target angles

11.4.5.2 `int main (int argc, char ** argv)`

11.4.6 Variable Documentation

11.4.6.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.4.6.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.4.6.3 `char const* __help__ = "> demo-benchmark --port=2 --dsaport=3 -v\n"`

11.4.6.4 `char const* __url__ = "http://www.schunk.com"`

11.4.6.5 `char const* __version__ = "$Id: demo-benchmark.cpp 10895 2013-11-11 14:15:08Z Osswald2 $"`

11.4.6.6 `cDBG cdbg(false,"red")`

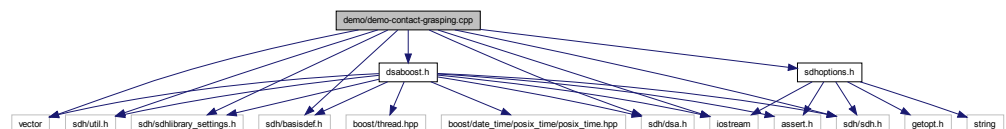
11.4.6.7 `char const* usage = "usage: demo-benchmark [options]\n"`

11.5 demo/demo-contact-grasping.cpp File Reference

Simple script to do grasping using tactile sensor info feedback. See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdh/dsa.h"
#include "sdhoptions.h"
#include "dsabooost.h"
```

Include dependency graph for demo-contact-grasping.cpp:



Functions

- cDBG [cdbg](#) (false,"red")
- void [GotoStartPose](#) (cSDH &hand, char const *msg)
- void [AxisAnglesToFingerAngles](#) (std::vector< double > aa, std::vector< double > &fa0, std::vector< double > &fa1, std::vector< double > &fa2)
- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-contact-grasping [options]\n"

Some informative variables

- char const * [__help__](#) = " > demo-contact-grasping --port=2 --dsaport=3 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-contact-grasping.cpp 10894 2013-11-11 13:58:38Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.5.1 Detailed Description

Simple script to do grasping using tactile sensor info feedback. See [__help__](#) and online help ("-h" or "--help") for available options.

11.5.2 General file information

Author

Dirk Osswald

Date

2009-08-02

11.5.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.5.4 Function Documentation

11.5.4.1 `void AxisAnglesToFingerAngles (std::vector< double > aa, std::vector< double > & fa0, std::vector< double > & fa1, std::vector< double > & fa2)`

11.5.4.2 `cDBG cdbg (false , "red")`

11.5.4.3 `void GotoStartPose (cSDH & hand, char const * msg)`

11.5.4.4 `int main (int argc, char ** argv)`

finished:

11.5.5 Variable Documentation

11.5.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.5.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.5.5.3 `char const* __help__ = " > demo-contact-grasping --port=2 --dsaport=3 -v\n"`

11.5.5.4 `char const* __url__ = "http://www.schunk.com"`

11.5.5.5 `char const* __version__ = "$Id: demo-contact-grasping.cpp 10894 2013-11-11 13:58:38Z Osswald2 $"`

11.5.5.6 `char const* usage = "usage: demo-contact-grasping [options]\n"`

11.6 demo/demo-dsa-simple.cpp File Reference

Simple program to test class cDSA (tactile sensor reading). See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include "sdh/sdhlibrary_settings.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dsa.h"
#include "sdh/basisdef.h"
#include "sdh/util.h"
```

Include dependency graph for demo-dsa-simple.cpp:

Functions

- int [main](#) (int argc, char **argv)

Variables

Some informative variables

Some definitions that describe the demo program

- char const * [__help__](#) = "parameters are fixed here in the source code\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-dsa-simple.cpp 12284 2014-09-30 08:28:44Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2014 SCHUNK GmbH & Co. KG"
- char const * [usage](#) = "usage: demo-dsa-simple\n"

11.6.1 Detailed Description

Simple program to test class cDSA (tactile sensor reading). See [__help__](#) and online help ("-h" or "--help") for available options.

11.6.2 General file information

Author

Dirk Osswald

Date

2014-04-03

11.6.3 Copyright

- Copyright (c) 2014 SCHUNK GmbH & Co. KG

11.6.4 Function Documentation

11.6.4.1 int main (int argc, char ** argv)

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

11.6.5 Variable Documentation

11.6.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.6.5.2 `char const* __copyright__ = "Copyright (c) 2014 SCHUNK GmbH & Co. KG"`

11.6.5.3 `char const* __help__ = "parameters are fixed here in the source code\n"`

11.6.5.4 `char const* __url__ = "http://www.schunk.com"`

11.6.5.5 `char const* __version__ = "$Id: demo-dsa-simple.cpp 12284 2014-09-30 08:28:44Z
Osswald2 $"`

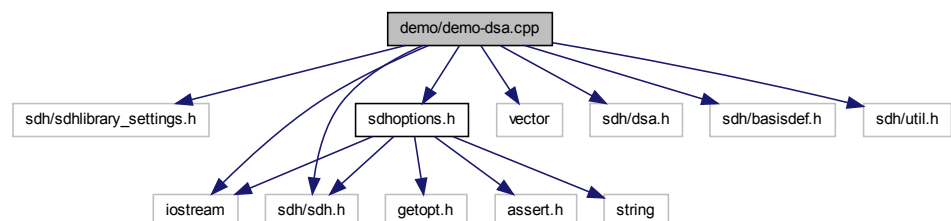
11.6.5.6 `char const* usage = "usage: demo-dsa-simple\n"`

11.7 demo/demo-dsa.cpp File Reference

Simple program to test class cDSA. See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include "sdh/sdhlibrary_settings.h"
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dsa.h"
#include "sdh/basisdef.h"
#include "sdh/util.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-dsa.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

Some informative variables

Some definitions that describe the demo program

- char const * [__help__](#) = " documentation\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-dsa.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2008 SCHUNK GmbH & Co. KG"
- char const * [usage](#) = "usage: demo-dsa [options]\n"

11.7.1 Detailed Description

Simple program to test class cDSA. See [__help__](#) and online help ("-h" or "--help") for available options.

11.7.2 General file information

Author

Dirk Osswald, Winfried Baum (IPA)

Date

2008-06-12

11.7.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.7.4 Function Documentation

11.7.4.1 `int main (int argc, char ** argv)`

11.7.5 Variable Documentation

11.7.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.7.5.2 `char const* __copyright__ = "Copyright (c) 2008 SCHUNK GmbH & Co. KG"`

11.7.5.3 `char const* __help__ = " documentation\n"`

11.7.5.4 `char const* __url__ = "http://www.schunk.com"`

11.7.5.5 `char const* __version__ = "$Id: demo-dsa.cpp 10351 2013-06-18 16:28:14Z Osswald2
$"`

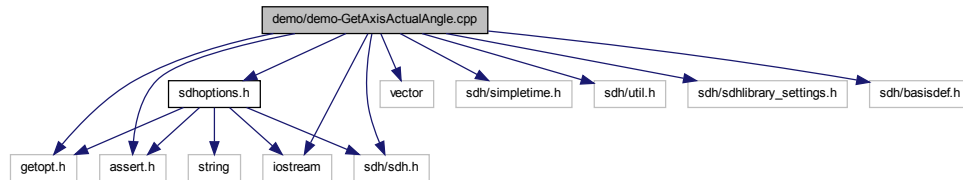
11.7.5.6 `char const* usage = "usage: demo-dsa [options]\n"`

11.8 demo/demo-GetAxisActualAngle.cpp File Reference

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/simpletime.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-GetAxisActualAngle.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-GetAxisActualAngle [options]\n"

Some informative variables

- char const * [__help__](#) = "> demo-GetAxisActualAngle --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-GetAxisActualAngle.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.8.1 Detailed Description

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

11.8.2 General file information

Author

Dirk Osswald

Date

2007-03-07

11.8.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.8.4 Function Documentation

11.8.4.1 `int main (int argc, char ** argv)`

11.8.5 Variable Documentation

11.8.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.8.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.8.5.3 `char const* __help__ = "> demo-GetAxisActualAngle --port=2 -v\n"`

11.8.5.4 `char const* __url__ = "http://www.schunk.com"`

11.8.5.5 `char const* __version__ = "$Id: demo-GetAxisActualAngle.cpp 10351 2013-06-18 16:28:14Z Osswald2 $"`

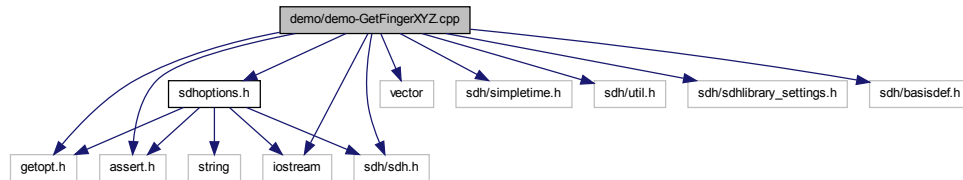
11.8.5.6 `char const* usage = "usage: demo-GetAxisActualAngle [options]\n"`

11.9 demo/demo-GetFingerXYZ.cpp File Reference

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/simpletime.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```


Include dependency graph for demo-GetFingerXYZ.cpp:



Functions

- int `main` (int argc, char **argv)

Variables

- char const * `usage` = "usage: demo-GetFingerXYZ [options]\n"

Some informative variables

- char const * `__help__` = " with MS Visual Studio\n"
- char const * `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * `__url__` = "http://www.schunk.com"
- char const * `__version__` = "\$Id: demo-GetFingerXYZ.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.9.1 Detailed Description

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See `__help__` and online help ("-h" or "--help") for available options.

11.9.2 General file information

Author

Dirk Osswald

Date

2007-03-07

Bug

When compiled with MS Visual Studio then using the "-R" command line parameter will make the demonstration program demo-GetFingerXYZ abort.

11.9.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.9.4 Function Documentation

11.9.4.1 `int main (int argc, char ** argv)`

11.9.5 Variable Documentation

11.9.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.9.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.9.5.3 `char const* __help__ = " with MS Visual Studio\n"`

11.9.5.4 `char const* __url__ = "http://www.schunk.com"`

11.9.5.5 `char const* __version__ = "$Id: demo-GetFingerXYZ.cpp 10351 2013-06-18 16:28:14Z Osswald2 $"`

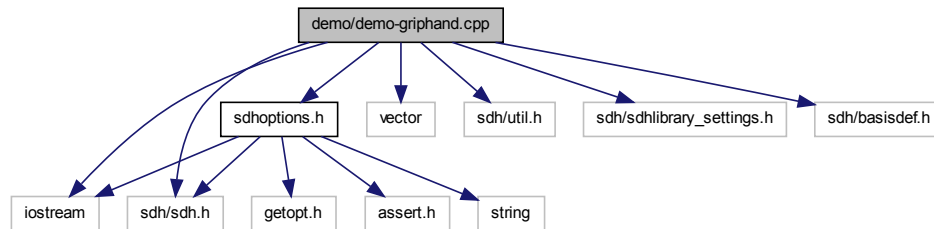
11.9.5.6 `char const* usage = "usage: demo-GetFingerXYZ [options]\n"`

11.10 demo/demo-griphand.cpp File Reference

Very simple demonstration program using the SDHLibrary-CPP: Demonstrate the use of the GripHand command See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-griphand.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-griphand [options]\n"

Some informative variables

- char const * [__help__](#) = "> demo-griphand --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-griphand.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.10.1 Detailed Description

Very simple demonstration program using the SDHLibrary-CPP: Demonstrate the use of the GripHand command See [__help__](#) and online help ("-h" or "--help") for available options.

11.10.2 General file information

Author

Dirk Osswald

Date

2009-12-04

Warning

The `cSDH::GripHand()` function is somewhat problematic (not interruptible), see its documentation.

11.10.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.10.4 Function Documentation

11.10.4.1 `int main (int argc, char ** argv)`

11.10.5 Variable Documentation

11.10.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.10.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.10.5.3 `char const* __help__ = "> demo-griphand --port=2 -v\n"`

11.10.5.4 `char const* __url__ = "http://www.schunk.com"`

11.10.5.5 `char const* __version__ = "$Id: demo-griphand.cpp 10351 2013-06-18 16:28:14Z Osswald2 $"`

11.10.5.6 `char const* usage = "usage: demo-griphand [options]\n"`

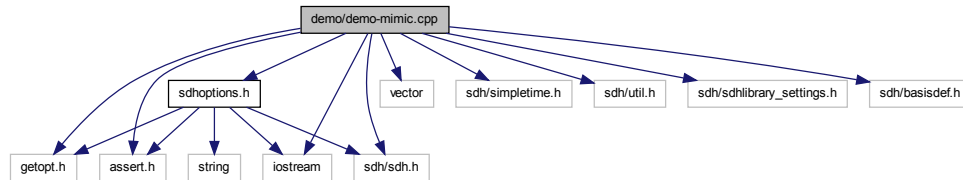
11.11 demo/demo-mimic.cpp File Reference

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/simpletime.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
```

```
#include "sdhoptions.h"
```

Include dependency graph for demo-mimic.cpp:



Functions

- cSDH * [GetHand](#) (cSDHOptions &options, cDBG &cdbg, int nb)
- int [main](#) (int argc, char **argv)

Variables

Some informative variables

- char const * [__help__](#) = "\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-mimic.cpp 6501 2011-03-01 17:35:45Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.11.1 Detailed Description

Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

11.11.2 General file information

Author

Dirk Osswald

Date

2007-03-07

11.11.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.11.4 Function Documentation

11.11.4.1 `cSDH* GetHand (cSDHOptions & options, cDBG & cdbg, int nb)`

11.11.4.2 `int main (int argc, char ** argv)`

11.11.5 Variable Documentation

11.11.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.11.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.11.5.3 `char const* __help__ = "\n"`

11.11.5.4 `char const* __url__ = "http://www.schunk.com"`

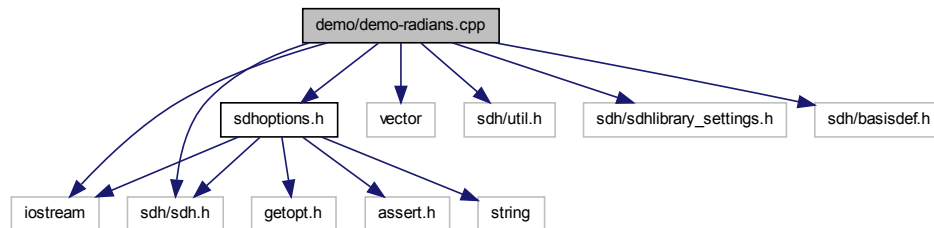
11.11.5.5 `char const* __version__ = "$Id: demo-mimic.cpp 6501 2011-03-01 17:35:45Z
Osswald2 $"`

11.12 demo/demo-radians.cpp File Reference

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control), commanding in radians. See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-radians.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-radians [options]\n"

Some informative variables

- char const * [__help__](#) = "> demo-radians --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-radians.cpp 11045 2013-11-27 15:12:49Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.12.1 Detailed Description

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control), commanding in radians. See [__help__](#) and online help ("-h" or "--help") for available options.

11.12.2 General file information

Author

Dirk Osswald

Date

2007-01-18

This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-*.cpp programms, or of course the html/pdf documentation.

11.12.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.12.4 Function Documentation

11.12.4.1 `int main (int argc, char ** argv)`

11.12.5 Variable Documentation

11.12.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.12.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.12.5.3 `char const* __help__ = "> demo-radians --port=2 -v\n"`

11.12.5.4 `char const* __url__ = "http://www.schunk.com"`

11.12.5.5 `char const* __version__ = "$Id: demo-radians.cpp 11045 2013-11-27 15:12:49Z
Osswald2 $"`

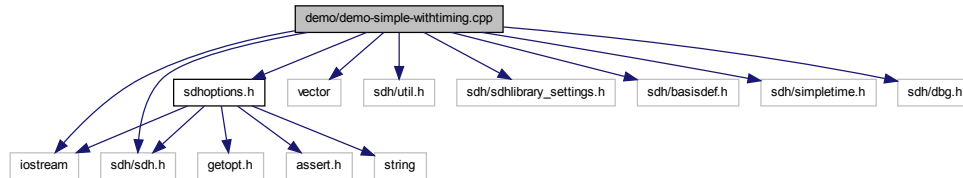
11.12.5.6 `char const* usage = "usage: demo-radians [options]\n"`

11.13 demo/demo-simple-withtiming.cpp File Reference

Very simple C++ programm to make an attached SDH move.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
#include "sdh/simpletime.h"
#include "sdh/dbg.h"
```


Include dependency graph for demo-simple-withtiming.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-simple-withtiming [options]\n"

Some informative variables

- char const * [__help__](#) = "Move proximal and distal joints of finger 1 three times by 10 degrees and measure time for these actions.\n(C++ demo application using the SDHLibrary-CPP library.)"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-simple-withtiming.cpp 6265 2010-12-02 13:59:45Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.13.1 Detailed Description

Very simple C++ programm to make an attached SDH move.

11.13.2 General file information

Author

Dirk Osswald

Date

2007-01-18

This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-*.cpp programmes, or of course the html/pdf documentation.

11.13.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.13.4 Function Documentation

11.13.4.1 `int main (int argc, char ** argv)`

11.13.5 Variable Documentation

11.13.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.13.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.13.5.3 `char const* __help__ = "Move proximal and distal joints of finger 1 three times by 10 degrees and measure time for these actions.\n(C++ demo application using the SDHLibrary-CPP library.)"`

11.13.5.4 `char const* __url__ = "http://www.schunk.com"`

11.13.5.5 `char const* __version__ = "$Id: demo-simple-withtiming.cpp 6265 2010-12-02 13:59:45Z Osswald2 $"`

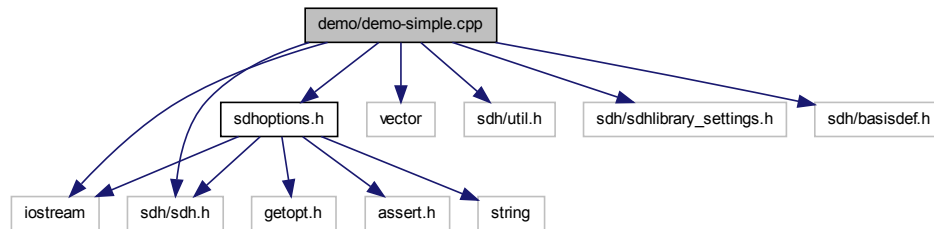
11.13.5.6 `char const* usage = "usage: demo-simple-withtiming [options]\n"`

11.14 demo/demo-simple.cpp File Reference

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-simple.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

- char const * [usage](#) = "usage: demo-simple [options]\n"

Some informative variables

- char const * [__help__](#) = "> demo-simple --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-simple.cpp 11045 2013-11-27 15:12:49Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.14.1 Detailed Description

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See [__help__](#) and online help ("-h" or "--help") for available options.

11.14.2 General file information

Author

Dirk Osswald

Date

2007-01-18

This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-*.cpp programm, or of course the html/pdf documentation.

11.14.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.14.4 Function Documentation

11.14.4.1 `int main (int argc, char ** argv)`

11.14.5 Variable Documentation

11.14.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.14.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.14.5.3 `char const* __help__ = "> demo-simple --port=2 -v\n"`

11.14.5.4 `char const* __url__ = "http://www.schunk.com"`

11.14.5.5 `char const* __version__ = "$Id: demo-simple.cpp 11045 2013-11-27 15:12:49Z
Osswald2 $"`

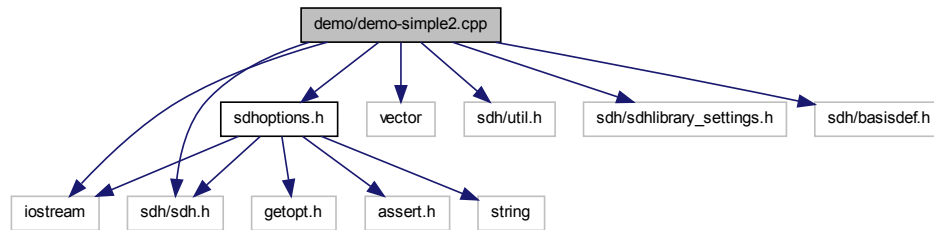
11.14.5.6 `char const* usage = "usage: demo-simple [options]\n"`

11.15 demo/demo-simple2.cpp File Reference

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Stop. See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-simple2.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

Some informative variables

- char const * [__help__](#) = "> demo-simple2 --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-simple2.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.15.1 Detailed Description

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Stop. See [__help__](#) and online help ("-h" or "--help") for available options.

11.15.2 General file information

Author

Dirk Osswald

Date

2007-01-18

This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-*.cpp programmes, or of course the html/pdf documentation.

11.15.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.15.4 Function Documentation

11.15.4.1 `int main (int argc, char ** argv)`

11.15.5 Variable Documentation

11.15.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.15.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.15.5.3 `char const* __help__ = "> demo-simple2 --port=2 -v\n"`

11.15.5.4 `char const* __url__ = "http://www.schunk.com"`

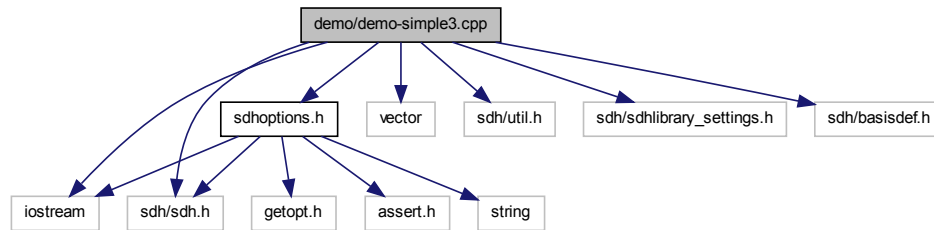
11.15.5.5 `char const* __version__ = "$Id: demo-simple2.cpp 10351 2013-06-18 16:28:14Z
Osswald2 $"`

11.16 demo/demo-simple3.cpp File Reference

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and WaitAxis. See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-simple3.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

Some informative variables

- char const * [__help__](#) = "> demo-simple3 --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-simple3.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.16.1 Detailed Description

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and WaitAxis. See [__help__](#) and online help ("-h" or "--help") for available options.

11.16.2 General file information

Author

Dirk Osswald

Date

2007-01-18

This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-*.cpp programm, or of course the html/pdf documentation.

11.16.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.16.4 Function Documentation

11.16.4.1 `int main (int argc, char ** argv)`

11.16.5 Variable Documentation

11.16.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.16.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.16.5.3 `char const* __help__ = "> demo-simple3 --port=2 -v\n"`

11.16.5.4 `char const* __url__ = "http://www.schunk.com"`

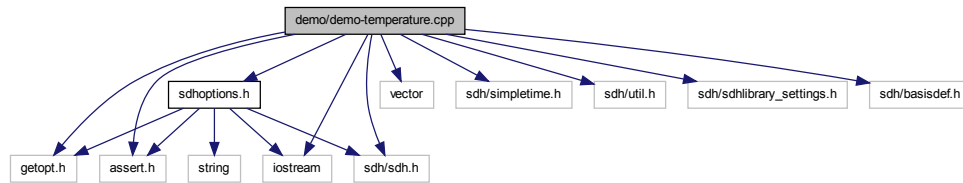
11.16.5.5 `char const* __version__ = "$Id: demo-simple3.cpp 10351 2013-06-18 16:28:14Z
Osswald2 $"`

11.17 demo/demo-temperature.cpp File Reference

Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/simpletime.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```


Include dependency graph for demo-temperature.cpp:



Functions

- int [main](#) (int argc, char **argv)

Variables

Some informative variables

- char const * [__help__](#) = "> demo-temperature --port=2 -v\n"
- char const * [__author__](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * [__url__](#) = "http://www.schunk.com"
- char const * [__version__](#) = "\$Id: demo-temperature.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * [__copyright__](#) = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.17.1 Detailed Description

Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See [__help__](#) and online help ("-h" or "--help") for available options.

11.17.2 General file information

Author

Dirk Osswald

Date

2007-01-18

11.17.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.17.4 Function Documentation

11.17.4.1 `int main (int argc, char ** argv)`

11.17.5 Variable Documentation

11.17.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.17.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.17.5.3 `char const* __help__ = "> demo-temperature --port=2 -v\n"`

11.17.5.4 `char const* __url__ = "http://www.schunk.com"`

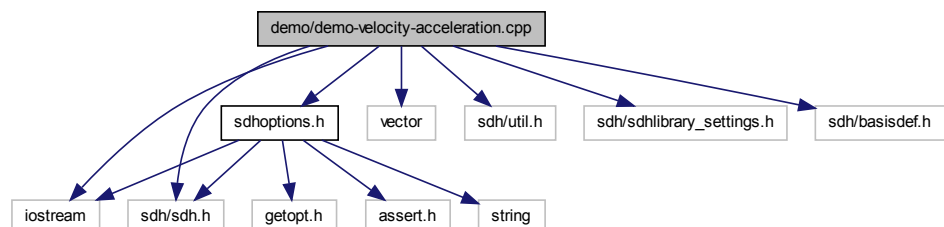
11.17.5.5 `char const* __version__ = "$Id: demo-temperature.cpp 10351 2013-06-18 16:28:14Z Osswald2 $"`

11.18 demo/demo-velocity-acceleration.cpp File Reference

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger in "velocity with acceleration ramp" control mode. See [__help_-__](#) and online help ("-h" or "--help") for available options.

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlbrary_settings.h"
#include "sdh/basisdef.h"
#include "sdhoptions.h"
```

Include dependency graph for demo-velocity-acceleration.cpp:



Functions

- int `main` (int argc, char **argv)

Variables

- char const * `usage` = "usage: demo-velocity-acceleration [options]\n"

Some informative variables

- char const * `__help__` = "> demo-velocity-acceleration --port=2 -v\n"
- char const * `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const * `__url__` = "http://www.schunk.com"
- char const * `__version__` = "\$Id: demo-velocity-acceleration.cpp 10351 2013-06-18 16:28:14Z Osswald2 \$"
- char const * `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.18.1 Detailed Description

Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger in "velocity with acceleration ramp" control mode. See `__help__` and online help ("-h" or "--help") for available options.

11.18.2 General file information

Author

Dirk Osswald

Date

2007-01-18

11.18.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.18.4 Function Documentation

11.18.4.1 `int main (int argc, char ** argv)`

11.18.5 Variable Documentation

11.18.5.1 `char const* __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`

11.18.5.2 `char const* __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

11.18.5.3 `char const* __help__ = "> demo-velocity-acceleration --port=2 -v\n"`

11.18.5.4 `char const* __url__ = "http://www.schunk.com"`

11.18.5.5 `char const* __version__ = "$Id: demo-velocity-acceleration.cpp 10351 2013-06-18 16:28:14Z Osswald2 $"`

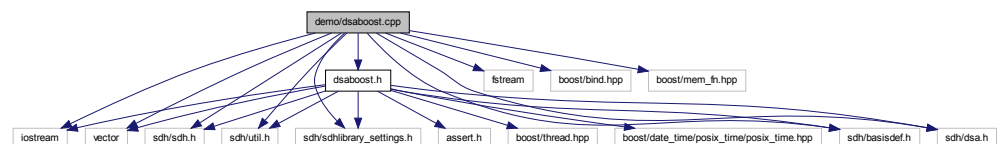
11.18.5.6 `char const* usage = "usage: demo-velocity-acceleration [options]\n"`

11.19 demo/dsaboost.cpp File Reference

helper stuff for the "boosted" DSA stuff

```
#include <iostream>
#include <fstream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdh/dsa.h"
#include "dsaboost.h"
#include <boost/bind.hpp>
#include <boost/mem_fn.hpp>
```

Include dependency graph for dsaboost.cpp:



Variables

- USING_NAMESPACE_SDH cDBG [cdbg](#)

11.19.1 Detailed Description

helper stuff for the "boosted" DSA stuff

11.19.2 General file information

Author

Dirk Osswald

Date

2009-08-02

11.19.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.19.4 Variable Documentation

11.19.4.1 USING_NAMESPACE_SDH cDBG [cdbg](#)

11.20 demo/dsaboost.h File Reference

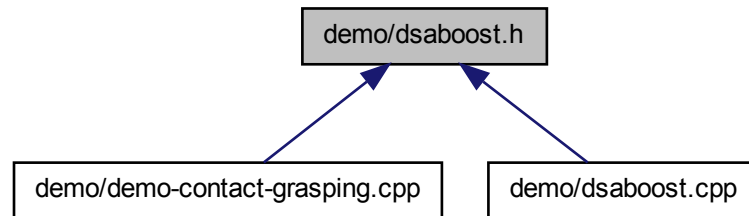
helper stuff for the DSA using boost

```
#include <iostream>
#include <vector>
#include <assert.h>
#include "sdh/sdh.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/basisdef.h"
#include "sdh/dsa.h"
#include "boost/thread.hpp"
#include "boost/date_time/posix_time/posix_time.hpp"
```

Include dependency graph for dsaboost.h:



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



Classes

- class [cDSAUpdater](#)

Class to create an updater thread for continuously updating tactile sensor data.

- class [cIsGraspedBase](#)

abstract base class for calculation of IsGrasped condition using tactile sensor information

- class [cIsGraspedByArea](#)

class for calculation of IsGrasped condition using an expected area of contact measured by the tactile sensors

11.20.1 Detailed Description

helper stuff for the DSA using boost

11.20.2 General file information

Author

Dirk Osswald

Date

2009-08-03

11.20.3 Copyright

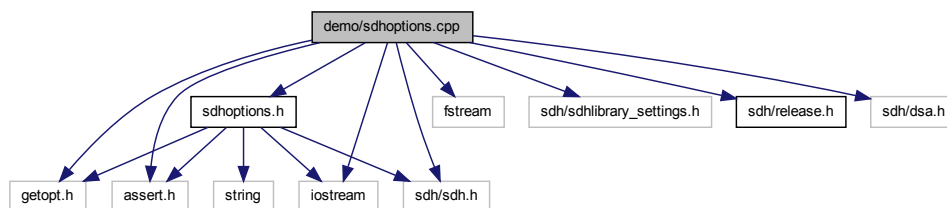
Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.21 demo/sdhoptions.cpp File Reference

Implementation of a class to parse common SDH related command line options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <fstream>
#include "sdh/sdh.h"
#include "sdh/sdhlbrary_settings.h"
#include "sdh/release.h"
#include "sdh/dsa.h"
#include "sdhoptions.h"
```

Include dependency graph for sdhoptions.cpp:

**Defines**

- #define [XSTRINGIFY\(_x\)](#) `STRINGIFY(_x)`

- `#define STRINGIFY(_s) #_s`
helper macro for XSTRINGIFY, see there

Variables

- static char const * `sdhusage_general` = " \n"
general options
- static char const * `sdhusage_sdhcom_serial` = " \n"
RS232 communication options.
- static char const * `sdhusage_sdhcom_common` = " Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232\n"
Common communication options.
- static char const * `sdhusage_sdhcom_esdcan` = ""
ESD CAN communication options.
- static char const * `sdhusage_sdhcom_peakcan` = ""
PEAK CAN communication options.
- static char const * `sdhusage_sdhcom_cancommon` = ""
Common CAN communication options.
- static char const * `sdhusage_sdhcom_tcp` = " \n"
TCP communication options.
- static char const * `sdhusage_sdhoother` = " \n"
Other options.
- static char const * `sdhusage_dsacom` = " \n"
DSA (tactile sensor) communication options.
- static char const * `sdhusage_dsaother` = " This option can be used multiple times.\n"
DSA (tactile sensor) other options.
- static char const * `sdhusage_dsaadjust` = " \n"
DSA (tactile sensor) adjustment options.
- static char const * `sdhoptions_short_options` = "hvVd:l:p:T:b:cn:e:w:RFt:q:r:fSCM:"

short command line options accepted by the cSDHOptions class
- static struct option `sdhoptions_long_options` []
long command line options accepted by the cSDHOptions class

11.21.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

Author

Dirk Osswald

Date

2008-05-05

11.21.2 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.21.3 Define Documentation

11.21.3.1 `#define STRINGIFY(_s) #_s`

helper macro for XSTRINGIFY, see there

11.21.3.2 `#define XSTRINGIFY(_x) STRINGIFY(_x)`

macro for stringification of `_x`

allows to stringify the **value** of a macro:

```
#define foo 4

STRINGIFY( foo ) // yields "foo"
XSTRINGIFY( foo ) // yields "4"
```

11.21.4 Variable Documentation

11.21.4.1 `struct option sdhoptions_long_options[] [static]`

long command line options accepted by the [cSDHOptions](#) class

11.21.4.2 `char const* sdhoptions_short_options = "hvVd:l:p:T:b:cn:e:w:RFt:q:r:fSCM:" [static]`

short command line options accepted by the [cSDHOptions](#) class

11.21.4.3 `char const* sdhusage_dsaadjust = "\n" [static]`

DSA (tactile sensor) adjustment options.

11.21.4.4 `char const* sdhusage_dsacom = "\n" [static]`

DSA (tactile sensor) communication options.

11.21.4.5 `char const* sdhusage_dsaothor = " This option can be used multiple times.\n" [static]`

DSA (tactile sensor) other options.

11.21.4.6 `char const* sdhusage_general = "\n" [static]`

general options

11.21.4.7 `char const* sdhusage_sdhcom_canccommon = "" [static]`

Common CAN communication options.

11.21.4.8 `char const* sdhusage_sdhcom_common = " Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232\n" [static]`

Common communication options.

11.21.4.9 `char const* sdhusage_sdhcom_esdcan = "" [static]`

ESD CAN communication options.

11.21.4.10 `char const* sdhusage_sdhcom_peakcan = "" [static]`

PEAK CAN communication options.

11.21.4.11 `char const* sdhusage_sdhcom_serial = "\n" [static]`

RS232 communication options.

11.21.4.12 `char const* sdhusage_sdhcom_tcp = "\n" [static]`

TCP communication options.

11.21.4.13 `char const* sdhusage_sdhother = "\n" [static]`

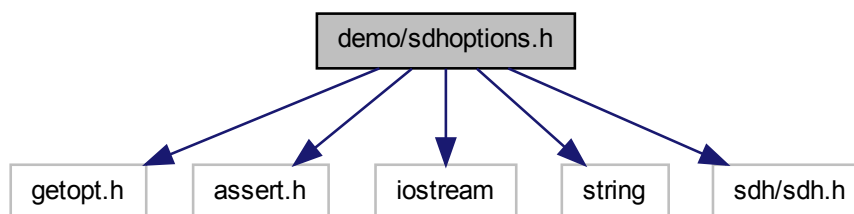
Other options.

11.22 demo/sdhoptions.h File Reference

Implementation of a class to parse common SDH related command line options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <string>
#include <sdh/sdh.h>
```

Include dependency graph for sdhoptions.h:



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



Classes

- class [cSDHOptions](#)
class for command line option parsing holding option parsing results

Defines

- #define [SDHUSAGE_DEFAULT](#) "general sdhcom_serial sdhcom_common sdhcom_esdcan sdhcom_peakcan sdhcom_cancancommon sdhcom_tcp"
- #define [SDH_DEFAULT_TCP_ADR](#) "192.168.1.42"
- #define [SDH_DEFAULT_TCP_PORT](#) 23
- #define [DSA_DEFAULT_TCP_PORT](#) 13000

11.22.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

11.22.2 General file information

Author

Dirk Osswald

Date

2008-05-05

11.22.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.22.4 Define Documentation

11.22.4.1 `#define DSA_DEFAULT_TCP_PORT 13000`

11.22.4.2 `#define SDH_DEFAULT_TCP_ADR "192.168.1.42"`

11.22.4.3 `#define SDH_DEFAULT_TCP_PORT 23`

11.22.4.4 `#define SDHUSAGE_DEFAULT "general sdhcom_serial sdhcom_common
sdhcom_esdcan sdhcom_peakcan sdhcom_cancommon sdhcom_tcp"`

string defining all the usage helptexts included by default

Bug

When compiled with VCC then the macros `WITH_ESD_CAN` / `WITH_PEAK_CAN` used above are not available since these are defined in the VCC project settings of the SDHLibrary VCC-Project. Therefore the value of `SDHUSAGE_DEFAULT` is incorrect and thus the [cSDHOptions](#) will display an incomplete usage string when called with `-h/--help`.

Workaround: use the online help contained in the doxygen documentation: [Online help of demonstration programs](#)

11.23 doc/onlinehelp-demo-benchmark.exe.dox File Reference**11.23.1 Detailed Description****11.24 doc/onlinehelp-demo-contact-grasping.exe.dox File Reference****11.24.1 Detailed Description****11.25 doc/onlinehelp-demo-dsa-simple.exe.dox File Reference****11.25.1 Detailed Description****11.26 doc/onlinehelp-demo-dsa-simple.log.dox File Reference****11.27 doc/onlinehelp-demo-dsa.exe.dox File Reference****11.27.1 Detailed Description****11.28 doc/onlinehelp-demo-GetAxisActualAngle.exe.dox File Reference****11.28.1 Detailed Description****11.29 doc/onlinehelp-demo-GetFingerXYZ.exe.dox File Reference****11.29.1 Detailed Description****11.30 doc/onlinehelp-demo-griphand.exe.dox File Reference****11.30.1 Detailed Description****11.31 doc/onlinehelp-demo-mimic.exe.dox File Reference****11.31.1 Detailed Description****11.32 doc/onlinehelp-demo-radians.exe.dox File Reference****11.32.1 Detailed Description****11.33 doc/onlinehelp-demo-ref.exe.dox File Reference**

Generated on Tue Sep 30 2014 15:59:02 for SDHLibrary-CPP by Doxygen

11.33.1 Detailed Description**11.34 doc/onlinehelp-demo-simple-withtiming.exe.dox File Reference****11.34.1 Detailed Description****11.35 doc/onlinehelp-demo-simple-withtiming.exe.dox File Reference**

11.41.1 Detailed Description

Doxyfile for generating documentation for SDHLibrary cpp using doxygen.

11.41.2 General file information

Author

Dirk Osswald

Date

2007-06-14

11.41.3 Links

- The online documentation for Doxygen can be found at <http://www.stack.nl/~dimitri/doxygen/>

11.41.4 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.42 Makefile File Reference

Makefile for [SDH](#) SDHLibrary C project.

11.42.1 Detailed Description

Makefile for [SDH](#) SDHLibrary C project.

11.42.2 General file information

Author

Dirk Osswald

Date

2007-01-03

This makefile can generate the C library itself, demo-programs and auxiliary stuff like doxygen documentation or generate a distribution for delivery to end users.

For a general description of the project see [general project information](#).

11.42.3 Makefile variables

The variables defined here state project specific settings which are then used by the goals and/or by the included, more generic sub makefiles like:

- Makefile-common
- Makefile-doc
- Makefile-rules

11.42.4 Makefile targets

- **all** : generate everything
 - **build** : generate library and demo programs
 - **doc** : generate all documentation
- **clean** : clean up generated program files, but not TAGS or doxygen doc
- **mrproper** : clean up all generated files, including TAGS and doxygen doc
- **tags** : generate emacs TAGS file

11.42.5 Links

- The online documentation for `gnu make` can be found at <http://www.gnu.org/software/make/>

11.42.6 Copyright

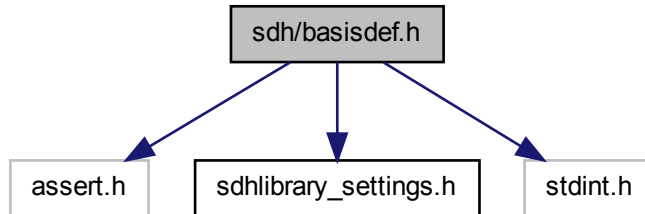
Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.43 sdh/basisdef.h File Reference

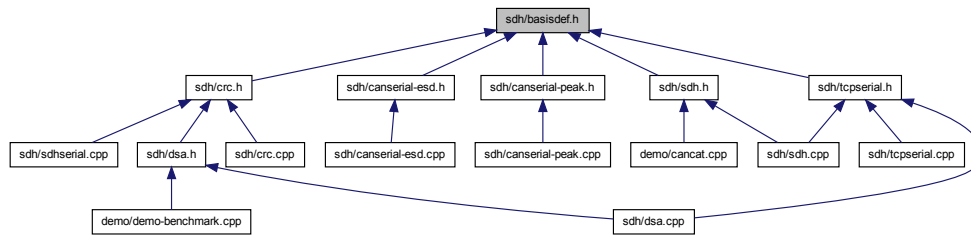
This file contains some basic definitions (defines, macros, datatypes)

```
#include <assert.h>
#include "sdhlibrary_settings.h"
#include <stdint.h>
```


Include dependency graph for basisdef.h:



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



Namespaces

- namespace [SDH](#)

Defines

- `#define` [SDH_ASSERT_TYPESIZES\(\)](#)
macro to assert that the defined typedefs have the expected sizes

Typedefs

- typedef `int8_t` [SDH::Int8](#)
signed integer, size 1 Byte (8 Bit)

- typedef uint8_t [SDH::UInt8](#)
unsigned integer, size 1 Byte (8 Bit)
- typedef int16_t [SDH::Int16](#)
signed integer, size 2 Byte (16 Bit)
- typedef uint16_t [SDH::UInt16](#)
unsigned integer, size 2 Byte (16 Bit)
- typedef int32_t [SDH::Int32](#)
signed integer, size 4 Byte (32 Bit)
- typedef uint32_t [SDH::UInt32](#)
unsigned integer, size 4 Byte (32 Bit)

11.43.1 Detailed Description

This file contains some basic definitions (defines, macros, datatypes)

11.43.2 General file information

Author

Jan Grewe, Dirk Osswald

Date

08.10.2004

- Datatypes: [SDH::Int8](#), [SDH::UInt8](#), [SDH::Int16](#), [SDH::UInt16](#), [SDH::Int32](#), [SDH::UInt32](#)

11.43.3 Copyright

Copyright (c) 2006 SCHUNK GmbH & Co. KG

11.43.4 Define Documentation

11.43.4.1 #define SDH_ASSERT_TYPESIZES()

Value:

```
do {
    assert( sizeof( Int8 )    == 1 );    \
    assert( sizeof( UInt8 )   == 1 );    \
    assert( sizeof( Int16 )   == 2 );    \
    assert( sizeof( UInt16 )  == 2 );    \
}
```

```

    assert( sizeof( Int32 )  == 4 );    \
    assert( sizeof( UInt32 ) == 4 );    \
} while (0)

```

macro to assert that the defined typedefs have the expected sizes

11.44 sdh/canserial-esd.cpp File Reference

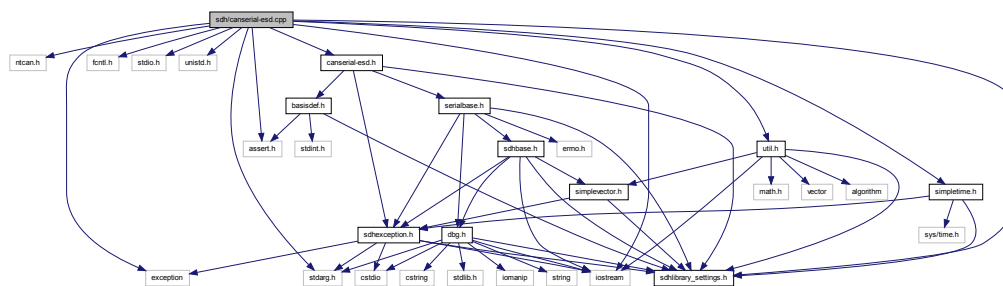
Implementation of class [SDH::cCANSerial_ESD](#), a class to access an ESD CAN interface on cygwin/linux and Visual Studio.

```

#include "ntcan.h"
#include "sdhlibrary_settings.h"
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <iostream>
#include <exception>
#include <stdarg.h>
#include <assert.h>
#include "canserial-esd.h"
#include "simpletime.h"
#include "util.h"

```

Include dependency graph for canserial-esd.cpp:



Classes

- class [SDH::cCANSerial_ESD_Internal](#)
internal hardware specific implementation details of the lowlevel ESD CAN interface

Namespaces

- namespace [SDH](#)

Defines

- #define [SDH_CANSERIAL_ESD_DEBUG](#) 1
- #define [DBG](#)(...)

Functions

- char const * [ESD_strerror](#) (NTCAN_RESULT rc)

11.44.1 Detailed Description

Implementation of class [SDH::cCANSerial_ESD](#), a class to access an ESD CAN interface on cygwin/linux and Visual Studio.

11.44.2 General file information

Author

Dirk Osswald

Date

2007-02-20

11.44.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.44.4 Define Documentation

11.44.4.1 #define [DBG](#)(...)

Value:

```
do {
    __VA_ARGS__;
} while (0)
```

instead of guarding every debug output with `#if SDH_CANSERIAL_ESD_DEBUG / #endif` we use this `DBG` macro that expands to a stream output to a `dbg` object or to `"`; depending on the value of `SDH_CANSERIAL_ESD_DEBUG`

11.44.4.2 #define SDH_CANSERIAL_ESD_DEBUG 1

Flag, if true then code for debug messages is included.

The debug messages must still be enabled at run time by setting the `some_cRS232_object.dbg.SetFlag(1)`.

This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

11.44.5 Function Documentation

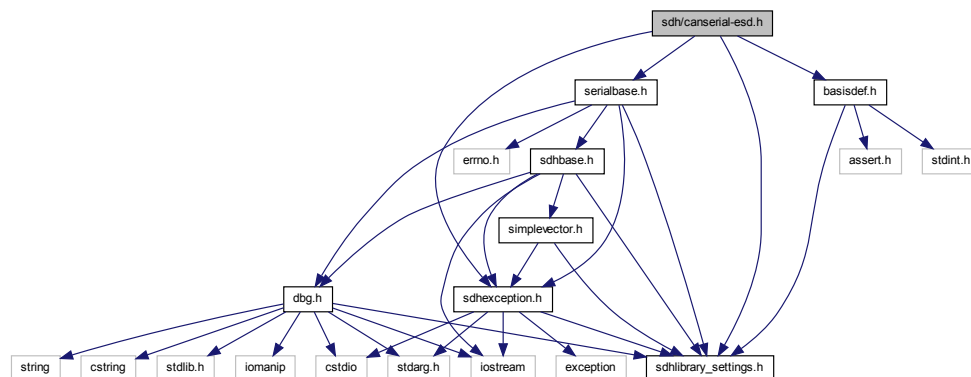
11.44.5.1 char const* ESD_strerror (NTCAN_RESULT rc)

11.45 sdh/canserial-esd.h File Reference

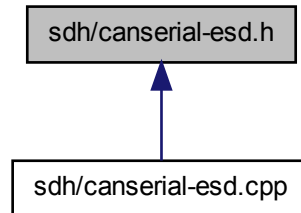
Interface of class [SDH::cCANSerial_ESD](#), class to access CAN bus via ESD card on cygwin/linux.

```
#include "sdhexception.h"
#include "serialbase.h"
#include "basisdef.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for canserial-esd.h:



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



Classes

- class [SDH::cCANSerial_ESDException](#)
Derived exception class for low-level CAN ESD related exceptions.
- class [SDH::cCANSerial_ESD](#)
Low-level communication class to access a CAN port from company ESD (<http://www.esd.eu/>)

Namespaces

- namespace [SDH](#)

Defines

- #define [CAN_ESD_TXQUEUEUSIZE](#) 32
transmit queue size for CAN frames
- #define [CAN_ESD_RXQUEUEUSIZE](#) 512
receive queue size for CAN frames

11.45.1 Detailed Description

Interface of class [SDH::cCANSerial_ESD](#), class to access CAN bus via ESD card on cygwin/linux.

11.45.2 General file information

Author

Dirk Osswald

Date

2008-05-02

11.45.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.45.4 Define Documentation

11.45.4.1 `#define CAN_ESD_RXQUEUESIZE 512`

receive queue size for CAN frames

11.45.4.2 `#define CAN_ESD_TXQUEUESIZE 32`

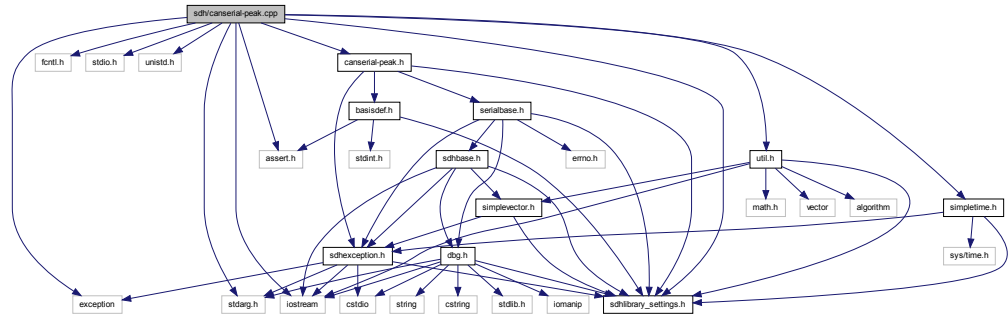
transmit queue size for CAN frames

11.46 sdh/canserial-peak.cpp File Reference

Implementation of class [SDH::cCANSerial_PEAK](#), a class to access a PEAK CAN interface on cygwin/linux and Visual Studio.

```
#include "sdhlibrary_settings.h"
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <iostream>
#include <exception>
#include <stdarg.h>
#include <assert.h>
#include "canserial-peak.h"
#include "simpletime.h"
#include "util.h"
```

Include dependency graph for canserial-peak.cpp:



Classes

- class [SDH::cCANSerial_PEAK_Internal](#)
internal hardware specific implementation details of the lowlevel PEAK CAN interface

Namespaces

- namespace [SDH](#)

Defines

- #define [USE_HANDLES\(H_\)](#)
- #define [USE_HANDLE\(H_\)](#)
- #define [SDH_CANSERIAL_PEAK_DEBUG](#) 1
- #define [DBG\(...\)](#)
- #define [M_CMSG_MSG\(\)](#) m_cmsg

Typedefs

- typedef void * [SDH::PCAN_HANDLE](#)
Linux libpcan uses HANDLE where Windows Pcan_usb.h uses no handle at all:

Functions

- char const * [PEAK_strerror](#) (DWORD rc)

11.46.1 Detailed Description

Implementation of class [SDH::cCANSerial_PEAK](#), a class to access a PEAK CAN interface on cygwin/linux and Visual Studio.

11.46.2 General file information

Author

Steffen Ruehl, Dirk Osswald

Date

2009-07-29

11.46.3 Define Documentation

11.46.3.1 `#define DBG(...)`

Value:

```
do {
    __VA_ARGS__;
} while (0)
```

instead of guarding every debug output with `#if SDH_CANSERIAL_PEAK_DEBUG` / `#endif` we use this `DBG` macro that expands to a stream output to a `dbg` object or to `","` depending on the value of `SDH_CANSERIAL_PEAK_DEBUG`

11.46.3.2 `#define M_CMSG_MSG() m_cmsg`

11.46.3.3 `#define SDH_CANSERIAL_PEAK_DEBUG 1`

Flag, if true then code for debug messages is included.

The debug messages must still be enabled at run time by setting the `some_cRS232_object.dbg.SetFlag(1)`.

This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

11.46.3.4 `#define USE_HANDLE(H_)`

11.46.3.5 `#define USE_HANDLES(H_)`

11.46.4 Function Documentation

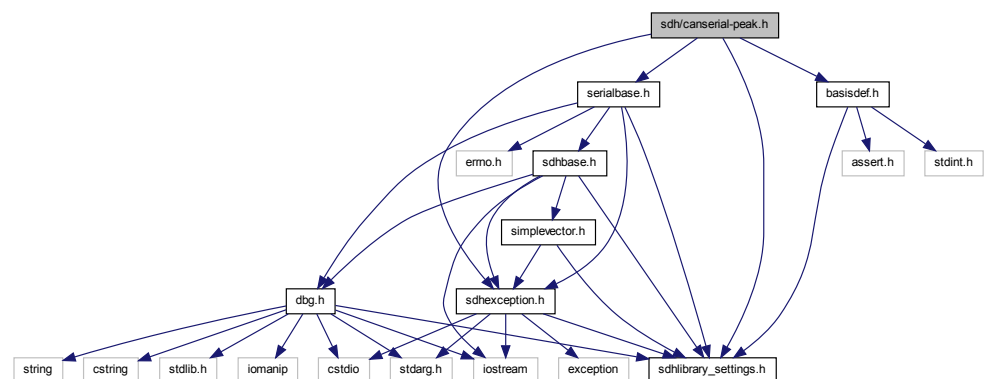
11.46.4.1 `char const* PEAK_strerror (DWORD rc)`

11.47 sdh/canserial-peak.h File Reference

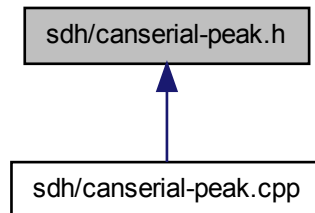
Interface of class `SDH::cCANSerial_PEAK`, class to access CAN bus via PEAK card on cygwin/linux.

```
#include "sdhexception.h"
#include "serialbase.h"
#include "basisdef.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for canserial-peak.h:



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



Classes

- class [SDH::cCANSerial_PEAKException](#)
Derived exception class for low-level CAN PEAK related exceptions.
- class [SDH::cCANSerial_PEAK](#)
Low-level communication class to access a CAN port from company PEAK (<http://www.peak-system.com>)

Namespaces

- namespace [SDH](#)

11.47.1 Detailed Description

Interface of class [SDH::cCANSerial_PEAK](#), class to access CAN bus via PEAK card on cygwin/linux.

11.47.2 General file information

Author

Steffen Ruehl, Dirk Osswald

Date

2009-07-29

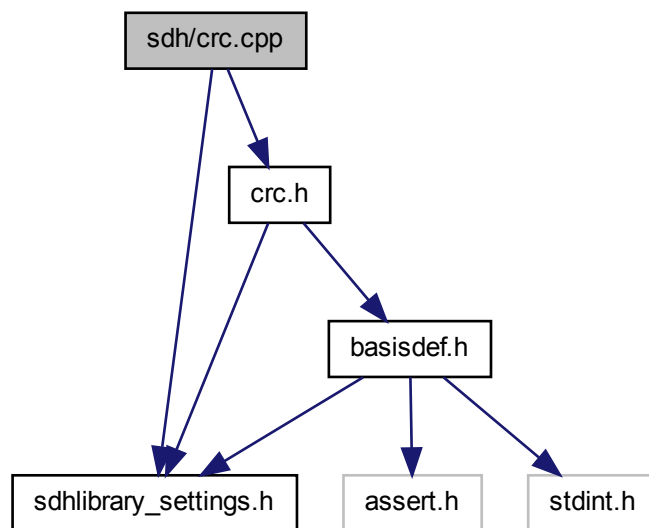
11.48 sdh/crc.cpp File Reference

Implementation of class [SDH::cCRC_DSACON32m](#) (actually only the static members all other is derived).

```
#include "sdhlibrary_settings.h"
```

```
#include "crc.h"
```

Include dependency graph for crc.cpp:



11.48.1 Detailed Description

Implementation of class [SDH::cCRC_DSACON32m](#) (actually only the static members all other is derived).

11.48.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.48.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

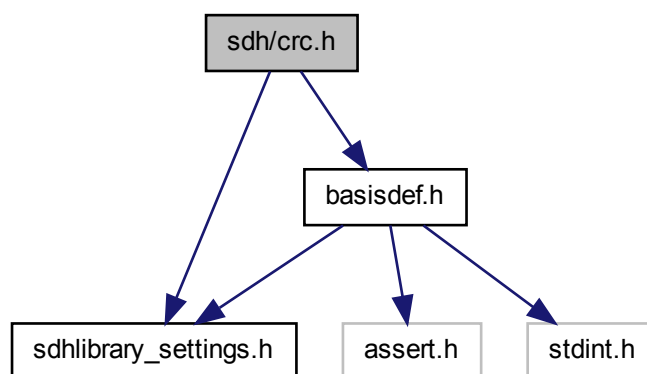
11.49 sdh/crc.h File Reference

This file contains interface to cCRC, a class to handle CRC calculation.

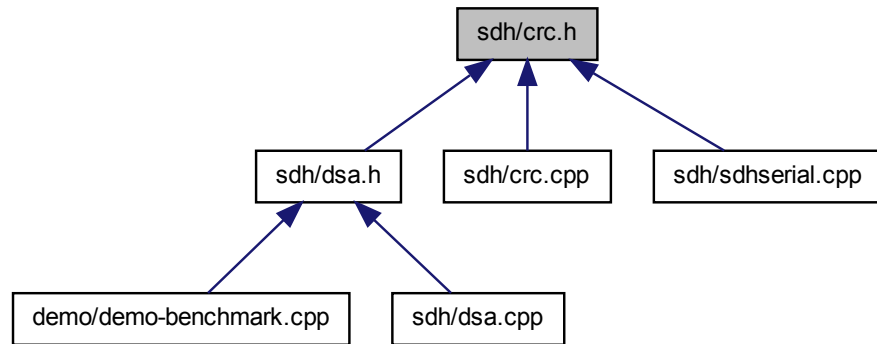
```
#include "sdhlibrary_settings.h"
```

```
#include "basisdef.h"
```

Include dependency graph for crc.h:



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



Classes

- class [SDH::cCRC](#)
Cyclic Redundancy Code checker class, used for protecting communication against transmission errors.
- class [SDH::cCRC_DSACON32m](#)
A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller.
- class [SDH::cCRC_SDH](#)
A derived CRC class that uses a CRC table and initial value suitable for protecting the binary communication with [SDH](#) via RS232.

Namespaces

- namespace [SDH](#)

Typedefs

- typedef UInt16 [SDH::tCRCValue](#)
the data type used to calculate and exchange CRC values with DSACON32m (16 bit integer)

11.49.1 Detailed Description

This file contains interface to cCRC, a class to handle CRC calculation.

11.49.2 General file information

Author

Dirk Osswald

Date

2008-06-09

11.49.3 Copyright

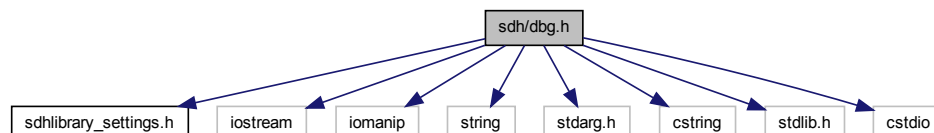
Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.50 sdh/dbg.h File Reference

This file contains interface and implementation of class [SDH::cDBG](#), a class for color-full debug messages.

```
#include "sdhlibrary_settings.h"
#include <iostream>
#include <iomanip>
#include <string>
#include <stdarg.h>
#include <cstring>
#include <stdlib.h>
#include <cstdio>
```

Include dependency graph for dbg.h:



- class `SDH::cDBG`
A class to print colored debug messages.
- class `SDH::cHexString`
dummy class for (debug) stream output of bytes as list of hex values

- namespace **SDH**

- `#define VAR(_d, _var) (_d) << #_var << "=" << _var << "\n"`
- `#define VAL(var) # var << "=" << var << " "`

- `VCC_EXPORT std::ostream & SDH::operator<< (std::ostream &stream, cHexString const &s)`
output the bytes in s to stream as a list of space separated hex bytes (without 0x prefix)

This file contains interface and implementation of class `SDH::cDBG`, a class for colorfull debug messages.

11.50.2 General file information

Author

Dirk Osswald

Date

2007-02-22

11.50.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.50.4 Define Documentation

11.50.4.1 `#define VAL(_var) #_var << "=" << _var << " "`

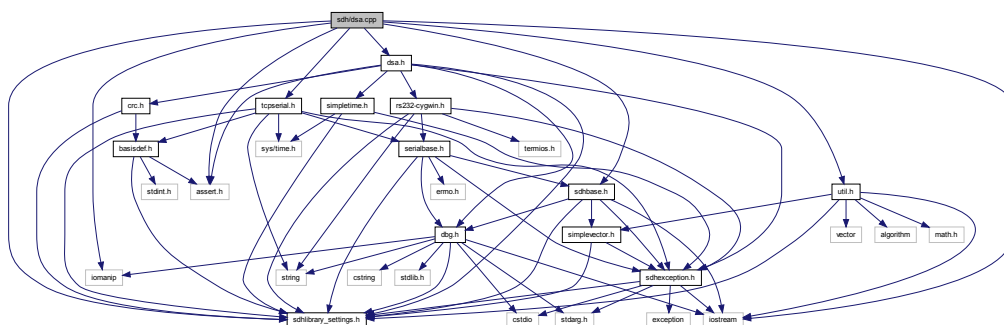
11.50.4.2 `#define VAR(_d, _var) (_d) << #_var << "=" << _var << "'\n"`

11.51 sdh/dsa.cpp File Reference

This file contains definition of [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <iostream>
#include <iomanip>
#include "dsa.h"
#include "sdhbase.h"
#include "util.h"
#include "tcpserial.h"
```

Include dependency graph for dsa.cpp:



Namespaces

- namespace **SDH**

Defines

- #define **PRINT_MEMBER**(_s, _var, _member) (_s) << " " << #_member << " = " << _var._member << "\n"
- #define **PRINT_MEMBER_HEX**(_s, _var, _member) (_s) << " " << #_member << " = 0x" << std::hex << int(var. member) << std::dec << "\n"

Enumerations

- enum eDSAPacketID {
eDSA_FULL_FRAME = 0x00, eDSA_QUERY_CONTROLLER_CONFIGURATION
= 0x01, eDSA_QUERY_SENSOR_CONFIGURATION = 0x02, eDSA_QUERY_
MATRIX_CONFIGURATION = 0x0B,
eDSA_CONFIGURE_DATA_ACQUISITION = 0x03, eDSA_QUERY_CONTROLLER_
FEATURES = 0x10, eDSA_READ_MATRIX_MASK = 0x04, eDSA_SET_
DYNAMIC_MASK = 0xAB,
eDSA_READ_DESCRIPTOR_STRING = 0x05, eDSA_LOOP = 0x06, eDSA_
QUERY_CONTROLLER_STATE = 0x0a, eDSA_SET_PROPERTIES_SAMPLE_
RATE = 0x0c,
eDSA_SET_PROPERTIES_CONTROL_VECTOR_FOR_MATRIX = 0x0d, eDSA_
GET_PROPERTIES_CONTROL_VECTOR_OF_MATRIX = 0x0e, eDSA_ADJUST_
MATRIX_SENSITIVITY = 0x0f, eDSA_GET_SENSITIVITY_ADJUSTMENT_
INFO = 0x12,
eDSA_SET_MATRIX_THRESHOLD = 0x13, eDSA_GET_MATRIX_THRESHOLD
= 0x14 }
}

Command ID for the DSA32m tactile sensor controller according to DSA32_-Command_Set_Reference_Manual.pdf.

Functions

- `std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sResponse const &response)`

11.51.1 Detailed Description

This file contains definition of `SDH::cDSA`, a class to communicate with the tactile sensors of the SDH.

11.51.2 General file information

Author

Dirk Osswald

Date

2008-06-09

11.51.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.51.4 Define Documentation

11.51.4.1 `#define PRINT_MEMBER(_s, _var, _member) (_s) << " " << #_member << "=" << _var._member << "\n"`

11.51.4.2 `#define PRINT_MEMBER_HEX(_s, _var, _member) (_s) << " " << #_member << "=0x" << std::hex << int(_var._member) << std::dec << "\n"`

11.51.5 Enumeration Type Documentation

11.51.5.1 enum eDSAPacketID

Command ID for the DSA32m tactile sensor controller according to DSA32_-Command_Set_Reference_Manual.pdf.

Enumerator:

eDSA_FULL_FRAME

eDSA_QUERY_CONTROLLER_CONFIGURATION

eDSA_QUERY_SENSOR_CONFIGURATION

eDSA_QUERY_MATRIX_CONFIGURATION

eDSA_CONFIGURE_DATA_ACQUISITION

eDSA_QUERY_CONTROLLER_FEATURES

eDSA_READ_MATRIX_MASK

eDSA_SET_DYNAMIC_MASK

eDSA_READ_DESCRIPTOR_STRING

eDSA_LOOP

eDSA_QUERY_CONTROLLER_STATE

eDSA_SET_PROPERTIES_SAMPLE_RATE

eDSA_SET_PROPERTIES_CONTROL_VECTOR_FOR_MATRIX

eDSA_GET_PROPERTIES_CONTROL_VECTOR_OF_MATRIX

eDSA_ADJUST_MATRIX_SENSITIVITY

eDSA_GET_SENSITIVITY_ADJUSTMENT_INFO

eDSA_SET_MATRIX_THRESHOLD

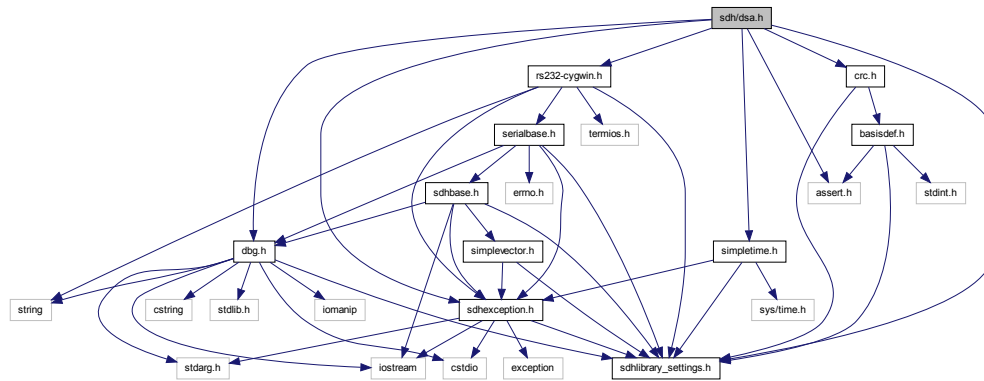
eDSA_GET_MATRIX_THRESHOLD

11.52 sdh/dsa.h File Reference

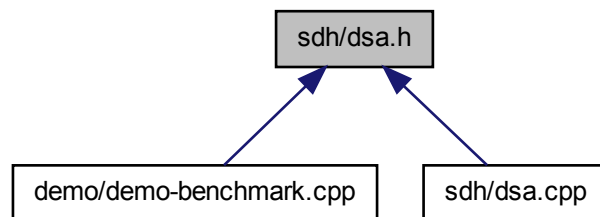
This file contains interface to [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include "sdhexception.h"
#include "dbg.h"
#include "rs232-cygwin.h"
#include "simpletime.h"
#include "crc.h"
```

Include dependency graph for dsa.h:



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



Classes

- class [SDH::cDSAException](#)
Derived exception class for low-level DSA related exceptions.
- class [SDH::cDSA](#)
[SDH::cDSA](#) is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH.
- struct [SDH::cDSA::sControllerInfo](#)
A data structure describing the controller info about the remote DSACON32m controller.

- struct [SDH::cDSA::sSensorInfo](#)
A data structure describing the sensor info about the remote DSACON32m controller.
- struct [SDH::cDSA::sMatrixInfo](#)
A data structure describing a single sensor matrix connected to the remote DSACON32m controller.
- struct [SDH::cDSA::sSensitivityInfo](#)
Structure to hold info about the sensitivity settings of one sensor patch.
- struct [SDH::cDSA::sTactileSensorFrame](#)
A data structure describing a full tactile sensor frame read from the remote DSACON32m controller.
- struct [SDH::cDSA::sContactInfo](#)
Structure to hold info about the contact of one sensor patch.
- struct [SDH::cDSA::sResponse](#)
data structure for storing responses from the remote DSACON32m controller

Namespaces

- namespace [SDH](#)

Defines

- #define [DSA_MAX_PREAMBLE_SEARCH](#) (2*3*(6*(14+13)) + 16)

Functions

- [sResponse](#) (UInt8 *_payload, int _max_payload_size)
constructor to init pointer and max size
- VCC_EXPORT std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sControllerInfo](#) const &controller_info)
- VCC_EXPORT std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sSensorInfo](#) const &sensor_info)
- VCC_EXPORT std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sMatrixInfo](#) const &matrix_info)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sResponse](#) const &response)
- VCC_EXPORT std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA](#) const &dsa)

Variables

- UInt16 [error_code](#)
0000h, if successful, otherwise error code
- UInt32 [serial_no](#)
- UInt8 [hw_version](#)
- UInt16 [sw_version](#)
- UInt8 [status_flags](#)
- UInt8 [feature_flags](#)
- UInt8 [senscon_type](#)
- UInt8 [active_interface](#)
- UInt32 [can_baudrate](#)
- UInt16 [can_id](#)
- UInt16 [nb_matrices](#)
- UInt16 [generated_by](#)
- UInt8 [hw_revision](#)
- float [texel_width](#)
- float [texel_height](#)
- UInt16 [cells_x](#)
- UInt16 [cells_y](#)
- UInt8 [uid](#) [6]
- UInt8 [reserved](#) [2]
- float [matrix_center_x](#)
- float [matrix_center_y](#)
- float [matrix_center_z](#)
- float [matrix_theta_x](#)
- float [matrix_theta_y](#)
- float [matrix_theta_z](#)
- float [fullscale](#)
- UInt8 [adj_flags](#)
- float [cur_sens](#)
- float [fact_sens](#)
- UInt8 [packet_id](#)
- UInt16 [size](#)
- UInt8 * [payload](#)
- Int32 [max_payload_size](#)

11.52.1 Detailed Description

This file contains interface to [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

11.52.2 General file information

Author

Dirk Osswald

Date

2008-06-09

11.52.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.52.4 Define Documentation

11.52.4.1 `#define DSA_MAX_PREAMBLE_SEARCH (2*3*(6*(14+13)) + 16)`

11.52.5 Function Documentation

11.52.5.1 `SDH_attribute_::sResponse (UInt8 * _payload, int _max_payload_size)`

constructor to init pointer and max size

11.52.6 Variable Documentation

11.52.6.1 `UInt8 active_interface`

11.52.6.2 `UInt8 adj_flags`

Bit vector indicating the sensitivity adjustment options:

- D7...2 reserved
- D1 1: user can change the sensitivity 0: sensitivity cannot be changed by the user

11.52.6.3 UInt32 can_baudrate**11.52.6.4 UInt16 can_id****11.52.6.5 UInt16 cells_x****11.52.6.6 UInt16 cells_y****11.52.6.7 float cur_sens**

Currently set sensitivity value. Floating point value. 0 is minimum sensitivity, 1.0 is maximum sensitivity.

11.52.6.8 UInt16 error_code

0000h, if successful, otherwise error code

11.52.6.9 float fact_sens

Sensitivity value that is used, if a factory restore command is issued. Floating point value. 0 is minimum sensitivity, 1.0 is maximum sensitivity.

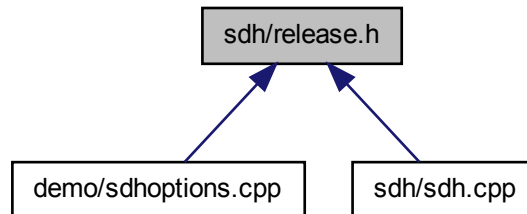
- 11.52.6.10 `UInt8 feature_flags`
- 11.52.6.11 `float fullscale`
- 11.52.6.12 `UInt16 generated_by`
- 11.52.6.13 `UInt8 hw_revision`
- 11.52.6.14 `UInt8 hw_version`
- 11.52.6.15 `float matrix_center_x`
- 11.52.6.16 `float matrix_center_y`
- 11.52.6.17 `float matrix_center_z`
- 11.52.6.18 `float matrix_theta_x`
- 11.52.6.19 `float matrix_theta_y`
- 11.52.6.20 `float matrix_theta_z`
- 11.52.6.21 `Int32 max_payload_size`
- 11.52.6.22 `UInt16 nb_matrices`
- 11.52.6.23 `UInt8 packet_id`
- 11.52.6.24 `UInt8* payload`
- 11.52.6.25 `UInt8 reserved[2]`
- 11.52.6.26 `UInt8 senscon_type`
- 11.52.6.27 `UInt32 serial_no`
- 11.52.6.28 `UInt16 size`
- 11.52.6.29 `UInt8 status_flags`
- 11.52.6.30 `UInt16 sw_version`
- 11.52.6.31 `float texel_height`
- 11.52.6.32 `float texel_width`
- 11.52.6.33 `UInt8 uid[6]`

11.53 sdh/release.h File Reference

Generated on Tue Sep 30 2014 15:59:02 for SDHLibrary-CPP by Doxygen

This file contains nothing but C/C++ defines with the name of the project itself ([PROJECT_NAME](#)) and the name of the release ([PROJECT_RELEASE](#)) of the whole project.

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



Defines

- #define `PROJECT_NAME` "SDHLibrary-CPP"
Name of the software project.
- #define `FIRMWARE_RELEASE_RECOMMENDED` "0.0.3.3"
- #define `PROJECT_RELEASE` "0.0.2.10"
Release name of the whole software project (a.k.a. as the "version" of the project).
- #define `PROJECT_DATE` "2014-02-28"
Date of the release of the software project.
- #define `PROJECT_COPYRIGHT` "(c) SCHUNK GmbH & Co. KG, 2007-2014"

11.53.1 Detailed Description

This file contains nothing but C/C++ defines with the name of the project itself (`PROJECT_NAME`) and the name of the release (`PROJECT_RELEASE`) of the whole project.

11.53.2 General file information

Author

Dirk Osswald

Date

2006-11-30

For a general description of the project see [general project information](#).

11.53.3 Copyright

Copyright (c) 2014 SCHUNK GmbH & Co. KG

11.53.4 Define Documentation

11.53.4.1 `#define FIRMWARE_RELEASE_RECOMMENDED "0.0.3.3"`

The recommended release of the firmware of an [SDH](#) used by this library

11.53.4.2 `#define PROJECT_COPYRIGHT "(c) SCHUNK GmbH & Co. KG, 2007-2014"`

11.53.4.3 `#define PROJECT_DATE "2014-02-28"`

Date of the release of the software project.

The date of the release of the project.

11.53.4.4 `#define PROJECT_NAME "SDHLibrary-CPP"`

Name of the software project.

The name of the "SDHLibrary-CPP" (C Library for accessing SDH from a PC) project.

11.53.4.5 `#define PROJECT_RELEASE "0.0.2.10"`

Release name of the whole software project (a.k.a. as the *"version"* of the project).

The release name of the "SDHLibrary-CPP" project. The doxygen comment below contains the changelog of the project.

A suffix of "-dev" indicates a work in progress, i.e. a not yet finished release. A suffix of "-a", "-b", ... indicates a bugfix release.

From newest to oldest the releases have the following names and features:

- **0.0.2.10:** 2014-02-28
 - bugfix for bug 1517: Bug: assertion failure in [util.cpp:207](#) in [NumerifyRelease\(\)](#)
- **0.0.2.9:** 2013-11-27
 - bugfix for bug 1462: Bug: limit checking is buggy when using radians
https://192.168.101.101/mechatronik/show_bug.cgi?id=1462
 - * fixed limit checking for angles, angular velocities and angular accelerations when using SDHLibrary with "Radians" as unit system, according to a bug report from Matei Ciocalie. Thanks.

- **0.0.2.8:** 2013-11-11
 - bugfix for bug 1440: Bug: communication via RS232 cannot be established from library due to write return -1 https://192.168.101.101/mechatronik/show_bug.cgi?id=1440
 - incorporating patches suggested from GeorgiaTech (for ESD-CAN on Linux)
- **0.0.2.7:** 2013-06-18
 - first version with recommended firmware 0.0.3.2 with ethernet TCP/IP support for tactile sensor data
- **0.0.2.6:** 2013-02-04
 - first version with recommended firmware 0.0.3.1 with ethernet TCP/IP support
- **0.0.2.5:** 2012-05-08
 - bugfixes 1153: Bug: Speicherzugriffsprobleme in [dsa.cpp](#) https://192.168.101.101/mechatronik/show_bug.cgi?id=1153
 - * fixed possible memory corruption issues according to a bug report from Fraunhofer IFF. Thanks.
- **0.0.2.4:** 2012-02-18
 - added bug description for 1020: [bug 1020 velocity control with acceleration ramp stops other axes when only a single one is addressed](#)
 - Enhancement Bug 1088: Task: Implement functions to [check compatibility of SDH firmware](#)
 - * added [SDH::cSDH::CheckFirmwareRelease\(\)](#) and [SDH::cSDH::GetFirmwareReleaseRecommended\(\)](#) to be able to check the actual versus the recommended [SDH](#) firmware release. Needed for the new [SDH](#) driver for ROS, see http://www.ros.org/doc/api/can_driver/can_driver.html
- **0.0.2.3:** 2011-05-11
 - working Bug 1012: Bug: cannot communicate with SDH via PEAK-CAN on Linux => testing compliance of SDHLibrary with PEAK CAN driver v7.1 for Linux
 - * cannot be confirmed as a bug
 - * minor bug in [cCANSerial_PEAK::Open\(\)](#) now only the read-ID is filtered instead of both write and read
 - * slightly improved [cconcat](#)
 - * improved debug output [cCANSerial_PEAK](#) and [cCANSerial_ESD](#) while
- **0.0.2.2:** 2011-04-27

- fixed Bug 1011: Bug: Binary communication does not work via RS232 on Linux The ISTRIP (Strip parity bits) bit MUST NOT be set on linux. If set then the MSbit of received bytes is set to 0 always... (Thanks to the hints of M. Schoepfer from Uni Bielefeld!)
- In distribution CAN bus and boost support is now disabled by default. (Can be enabled in Makefile-settings, see WITH_ESD_CAN, WITH_PEAK_CAN, HAVE_BOOST)

- **0.0.2.1:** 2011-03-09

- fixed Bug 844: Bug: Korean windows cannot handle specific unit characters like ĩȳ and ĩȳ The non ASCII string literals for degrees and squared were replaced by ASCII literals
- added Bug 996: Task: Version numbering of DSA32m firmware has changed since 2011-02-15 software version numbers for DSA32m are reported correctly for the new firmwares
- fixed Bug 703: Bug: Tactile sensor frames cannot be read reliably in single frame mode
 - * Some firmware versions of the DSA32m are not able to do single frame acquisition and enter push-mode. This might fill up the RS232 input buffer and leads to problems like reading of outdated frames or frame loss.
 - * Added workaround to stop push mode immediately after it was entered unintentionally
 - * made debug output of communication functions (RS232 and CAN and TCP) more concise
- Enhancement Bug 985: Task: Get Visual Studio to compile SDHLibrary-c++ as DLL
 - * The Visual Studio projects now generate and use a DLL version of the SDHLibrary

- **0.0.2.0:** 2011-02-08

- added new documentation file SDH2_exchanging_tactile_sensor_controller
- added support for communication via TCP (requires at least SDH firmware 0.0.3.0)
 - * enhancement Bug 874: Task: Enable TCP communication in SDHLibrary
 - * refactoring Bug 918: Task: Refactor generation of error messages
- added properties for "Release" configuration in VCC project files
- bugfix Bug 959: Bug: SDHLibrary does not compile in new cygwin-1.7 environment
 - * refactored device driver specific CAN stuff from headers into code files

- added demo-benchmark for measuring communication performance [Bug 963: Task: implement benchmark for measuring communication rate](#)
 - added support for fast binary communication with CRC protection [Bug 964: Task: implement new binary communication in SDHLibrary](#)
 - added commands to set target angles/velocities and get actual angles/velocities in one command, see `cSDH::SetAxisTargetGetAxisActualAngle()` and `cSDH::SetAxisTargetGetAxisActualVelocity()`
 - **These last two points along with the accompanying changes in the firmware boost the communication speed significantly!**
 - * For RS232 the communication cycles per second rise from ca. 30 to 75 Hz (according to demo-benchmark)
 - * For CAN the improvement goes even higher to ca. 160 Hz (according to demo-benchmark)
 - Doxygen documentation is now generated with doxygen-1.7.3 with included javascript search engine
- **0.0.1.18:**
 - added bug description for [Bug 822: Online help of C++ demo programs compiled with VCC misses description for CAN parameters](#)
 - added new documentation file `SDH2_FingerDimensions`
 - **0.0.1.17:** (2010-05-11)
 - enhancement [Task: Reorganise Makefiles to simplify making of user provided demo-programs](#) extracted common Makefile settings to Makefile-settings for better maintenance and extensibility,
 - Enhancement made acquiring of single tactile sensor frames available
 - **0.0.1.16:** (2010-04-12)
 - bugfix: [Bug 680: cDSA fails to communicate with Release 276 of DSACON32m Firmware](#)
 - **0.0.1.15:** (2010-03-05)
 - enhancement: [Bug 482: add support for adjusting sensitivity and threshold in SDHLibrary-CPP](#)
 - * added `cDSA.SetMatrixSensitivity()` `cDSA.GetMatrixSensitivity()` `cDSA.SetMatrixThreshold()` `cDSA.GetMatrixThreshold()`
 - * the sensitivity and the threshold can now be changed and saved temporarily or persistently to configuration memory of the DSACON32m
 - * added command line options to `demo-dsa` to adjust matrix sensitivity and threshold (needs DSACON32m Firmware \geq R268)

- **0.0.1.14:** (2010-02-19)
 - added documentation file SDH2_exchanging_tactile_sensors.pdf to distribution
 - added documentation file SDH2_wiring_test_board.pdf to distribution
 - added documentation file [SDH](#) training.pdf to distribution
 - updated documentation file SDH2_configuration-and-update.pdf
- **0.0.1.13:** (2010-02-02)
 - bugfix (firmware 0.0.2.10): [Bug 630: Bug: setting of pid parameters does not work](#)
- **0.0.1.12:** (2009-12-04)
 - bugfix: [Bug 575: grip related commands do not work properly](#)
 - * the actual answers for selgrip/grip from the [SDH](#) were different from the expected ones in sdhserial
 - * there was a problem with the read ahead data in readline in serialbase
 - bugfix: [Bug 470: made demo-contact-grasping.cpp actually work](#)
- **0.0.1.11:** (2009-12-01)
 - added support for CAN devices from PEAK, kindly provided by Steffen Ruehl and Zhixing Xue
 - * this caused some changes and additions in the command line option handling
 - * made the "Linux-only" PEAK port work on Windows as well
 - fixed bug in both ESD and PEAK CAN support that might prevent proper access to several SDHs on same CAN interface from the same process. (Problem was static data in communication classes)
 - added online help of demonstration programs to doxygen documentation
 - changed Makefile to use 'uname' instead of unreliable OSTYPE to determine OSNAME
 - added demo-contact-grasping to show how to do very basic reactive grasping (using boost::threads to evaluate tactile sensor data concurrently)
 - added demo-mimic to show how to access 2 SDHs, one mimicing the movements of the other
 - bugfix: [Bug 561: SDHLibray does not compile with VCC](#)
 - * the pragma pack was not used correctly, i.e. packing was not reset to normal after the structs to pack tightly
 - bugfix: [Bug 568: option -M causes segfault on demo-dsa](#)
 - enhancement: [Bug 480: Include DSACON32 firmware within distribution](#)

- **0.0.1.10:** (unofficial release 2009-10-05)

- added troubleshooting section to the CD contents in troubleshooting.html
- corrected checking of environment variable "OS" for the automatic disabling of the use of color in output. OS is "WINNT" on German windows XP, but "Windows_NT" on US-english Windows XP... Phhh
- while fixing **Bug 452: current demo-gui.py fails when connected to an old SDH v0.0.2.0** added firmware version specific code to make `cSDH::GetAxisLimitVelocity()` and `cSDH::GetAxisLimitAcceleration()` work on older firmwares
- made all operator<< receive a const reference as parameter. This should improve performance especially for the `cDBG::operator<<`
- bugfix: trying to connect to the tactile sensors of an **SDH** that is missing or switched off caused an infinite retry loop
- Bugfix: **Bug 351: make does not work recursively** reopen and fixed again since still had problems on Linux where the shell command "test" or "[" is somewhat picky about comparing strings ("==" is not understood)
- bugfix: **Bug 471: Incorporate feedback for 64 bit compatibility** changes to make compilation work on 64 bit systems according to feedback from Niklas Bergstroem.
- enhancement: Added possibility to set the RS232 device name format string (e.g. to use /dev/ttyUSBd like devices)
- bugfix: **Bug 469: make clean funktioniert unter Linux nicht**
- enhancement: **Bug 472: Objects of type cDSA should be able to recover from power cycling** There is now a `cDSA::Open()` member to reopen the connection to the DSA controller after a failure
- bugfix: https://192.168.101.101/mechatronik/show_bug.cgi?id=478>Bug 478: Online help of C++ demo programs is incorrect
- modified Makefile-doc, Makefile, Doxyfile to be able to use target specific variables to exclude/include files from documentation depending on whether internal or external docu is generated

- **0.0.1.9:** 2009-06-17

- added missing includes of `cstdio` for gcc-4.4 as reported by Hannes Saal.
- while adding webcheck to check the generated html files:
 - * corrected settings in Doxyfile so that tagfiles not available for customer are no longer used in distribution
 - * corrected broken/missing links in the distribution html files according to webcheck
- added forgotten demo-velocity-acceleration project for VCC

- bugfix: [Bug 433: Invalid negative velocities remain set when switching from speed based controllers back to pose controller](#) Adjusted documentation for SetController() accordingly
 - adjusted Library for new behaviour of firmware 0.0.2.7 in eCT_VELOCITY_-ACCELERATION controller type:
 - * acceleration must no longer be given with correct sign. The sign of the acceleration is now determined automatically from the signs and magnitudes of the current reference velocity and the target velocity
 - * Adjusted WaitAxis() since the state is now reported correctly by the firmware, even if in a speed based controller mode
 - * adjusted doxygen documentation and [demo-velocity-acceleration.cpp](#)
 - * current controller_type is now cached in cSDH object
 - * Now using the same acceleration limits as the firmware
 - Date of library is now reported as well for option -v in the demo programs
 - corrected doxygen description of GetTemperature()
 - enhancement: [Bug 442: acceleration limits cannot be queried from the firmware](#)
 - * added new commands to read acceleration limits from firmware
 - bugfix: [Bug 432: temperature output does not respect the -F Fahrenheit switch from the command line](#)
 - enhancement: updated / corrected doxygen comments
 - * updated known bugs
 - * guarded text "SDH" with "%SDH" in doxygen comments to prevent doxygen from auto-linking to [SDH](#) namespaceup
- **0.0.1.8:** 2009-05-18
 - enhancement: [Enhancement 263: Provide access to speed controller of SDH joints](#)
 - * provide access to the 2 additional controller types eCT_VELOCITY and eCT_VELOCITY_ACCELERATION which are provided by the firmware 0.0.2.6
 - * added new command cSDH.GetAxisReferenceVelocity() to access the internal reference velocity of the eCT_VELOCITY_ACCELERATION controller type
 - * added new demonstration script [demo-velocity-acceleration.cpp](#)
 - * the allowed lower limits for velocity and acceleration now have to be adjusted when the controller type changes.
 - enhancement [Enhancement 384: Make doxygen documentation match for SDHLibrary-CPP and SDHLibrary-python](#)
 - * added group with all demonstration programs
 - * included description of all known bugs
 - **0.0.1.7:** 2009-05-05

- Enhancement: added CAD Datafiles to distribution CD
- Bugfix: Bug 341: new gcc-4.x capable Makefile does not work in Linux Linux gcc does not know option --enable-auto-import, added cygwin specific code to Makefile
- Bugfix: Bug 351: make does not work recursively building the library did not work in one go since make did not work recursively due to errors in Makefile-subdir PIPESTATUS was used which is an automatic variable of the bash shell. So if another shell is used then PIPESTATUS cannot be used to determine if the recursively called make succeeded. Therefore we now export PIPESTATUS="0" (which means 'true' for shells) to keep make going on non bash shells.
- bugfix: Bug 342: RS232 communication does not work any more when compiled with MS-Visual Studio on Windows Resolved, parameters like bits per byte were not set explicitly, so the code worked with some interfaces only, depending on the default settings of the interface
- bugfix: Bug 361: No proper RS232 communication in C++ version of SDHLibrary Resolved, timeout handling improved
- bugfix: Bug 364: Remove ancient, no current stuff from the distribution; Separate build target into build_lib build_demo (and build_test)
- bugfix: Bug 366: demo-dsa.exe does not respect --debuglog
- bugfix: corrected output of internal datatypes members of type UInt8. These are now printed as hex instead of as char
- enhancement: debug outputs of low level communication in rs232-cygwin.cpp and rs232-vcc.cpp can be included / excluded at compile time using the SDH_RS232_CYGWIN_DEBUG / SDH_RS232_VCC_DEBUG macros. If included the debug messages can be enabled / disabled at runtime with the usual -d LEVEL parameter. (Level 4 is needed for these in the demos)
- bugfix: Bug 372: Disabled colored debug output on windows consoles for better readability
- bugfix: Bug 370: First call to demo-dsa after powering SDH fails The problem is related to the tactile sensor controller DSACON32m within the SDH. This controller needs approximately 8 seconds to "boot" up, and during that time it will not answer to requests from the SDHLibrary. Unfortunately the timeout mechanism of the SDHLibrary for Windows also had a bug which then made it wait forever for answers that would never come.
- bugfix: Bug 373: EmergencyStop did not work Fixed (copy & paste error while porting from python to C++)
- bugfix: Bug 379: invalid command line parameters cause segv in demo-dsa Fixed, missing ':' in option definition string
- bugfix: Bug 386: demo-simple2.cpp does not behave as expected Fixed, some code was commented out
- velocity and acceleration for virtual axis is no longer limited to [0..0], since that makes SetAxisTargetVelocity(All, x) and SetAxisTargetAcceleration(All, x) invalid for all x != 0.0

- **0.0.1.6:**

- extracted generation of distribution stuff from Makefile to Makefile-dist (since not needed by customer)
- corrected use of CC and CPPC variables in Makefiles. Now these can be set from the environment. Needed to test compilation with alternative compilers like gcc-4.x instead of std gcc-3.x
- corrected copy & paste errors in doxygen comments of cSDH::GetAxisMaxVelocity()
- Bugfix [Bug 333: Invalid default parameters and documentation for OpenESD_CAN](#) Mix-up of hex and dec representation of the default IDs used for CAN communication
- made compilation work without errors and warnings with gcc-4.3.2:
 - * added additional includes like "cstring"
 - * changed many char* to char const* to get rid of deprecation warnings
- changed generation of distribution:
 - * modified Doxyfile is now included so that user can generate documentation by himself
 - * doc target is no longer a subtarget of all target in distributed Makefile (user will most likely not want to regenerate the docu)
- updated links to misc packages and Weiss documentation in index-overview.html in distribution

- **0.0.1.5:** 2009-02-11

- bugfix: [Bug 322: AxisTargetVelocity cannot be set higher than 100 deg/s](#)
- bugfix: [Bug 323: SDHLibrary exceptions not deleted correctly when caught and handled](#)
- enhancement: [Enhancement 315: Add documentation files to distribution](#)

- **0.0.1.4:**

- bugfix: [Bug 267: SDHlib cannot be compiled on Linux](#)

- **0.0.1.3:** 2008-10-16

- bugfix: [Bug 266: demo-dsa does not work in the VCC version](#)

- **0.0.1.2:** 2008-10-14

- bugfix: target concat is now only added in the Makefile if WITH_ESD_CAN is 1
- bugfix: corrected copy/paste error: removed class qualifiers from member declarations since newer gccs do not accept fully qualified member names within class declarations (like aClass::aMember())

- bugfix: corrected parameter checking of cSDHSerial::vp()
- bugfix: corrected error in `apply()` in `util.h` (correct result was calculated only locally)
- made generation of Doxygen-doku work with Doxygen v1.5.5
- fixed bug in option detection
- made output of version info more verbose (with SOC and dates)
- implemented `cRS232::Read()`
- made `dsa.cpp/h` work with VCC
- corrected TCP calculation: corrected limb lengths and changed coordinate system from left to right handed
- added `cSDH::GetInfo`
- added `cSDHSerial::vlim()` and `cSDH::GetAxisLimitVelocity()` to read velocity limits
- bugfix: Bug 134: cannot generate doxygen documentation if SDH namespace is used
- bugfix: Bug 261: Header file problems with SDHLibrary-CPP on Linux
- bugfix: Bug 260: CAN access problems with SDHLibrary-CPP on Linux
- enhancement: Enable debugging to logfile in SDHLibrary-cpp
- enhancement: reduced overhead in `cDBG::operator<<`, no more searching for color strings on each call
- added demo-simple-withtiming to perform some simple OS-level time measurement
- change: changed parent class of `cSerialBaseException` to `cSDHErrorCommunication` to make the hierarchy more consistent

• 0.0.1.1:

- added `cancat` program for sending inaccessible commands like `change_rs232` via CAN
- added info commands corresponding to new info commands in firmware:
 - * in `sdhserial`:
 - `soc` - to read the SoC ID
 - `soc_date` : to read the date string of the SoC
 - `ver_date` : to read the release date of the firmware
 - * in `sdh`:
 - enhanced `GetInfo` to read all the above also
 - * in the option parser in `auxilliary` all the info is now printed if "-v" is given
- added `GetDuration` command: returns the calculate duration of the currently configured movement (target angle, velocity, acceleration, velocity profile) but does not execute the movement. This simplifies scripts like `sdhrecord.py` a great deal.

- made py.test test_sdh work again.
- while working on [Bug 224: Positioning delay of 5s when using SDHLibrary-CPP](#)
 - * removed call to [SleepSec\(\)](#) in loop in serial for VCC, needed to get rid of delays
 - * implemented cSimpleTime in VCC version
- added support for new firmware v0.0.2.0 commands
 - * soc, soc_date, ver_date
 - * GetDuration
- **0.0.1.0:** 2008-06-13
 - added basic support for the tactile sensors,
 - * new library classes cDSA, cCRC
 - * new demo program demo-dsa
- **0.0.0.9:** 2008-06-06
 - added missing files for vcc compilation in distribution
 - removed error in overloaded Sleep function. Internal version renamed to SleepSec.
 - made demo-GetAxisActualAngle and demo-temperature work in periodic mode in Visual Studio
 - untabified all source and header files for use with Visual Studio
 - autostart feature for distribution CD
- **0.0.0.8-a:**
 - added project files for the demo-* programs to the Visual Studio solutions file to make the demo programs available under VCC too
 - added forgotten WITH_ESD_CAN=1 to Visual Studio project file for SDHLibrary
 - workaround for accessing RS232 from VCC (WriteFile() does not return number of bytes sent)
 - With VCC the communication via RS232 still has some bugs: long pauses and timeouts,
 - With VCC the "-t" parameter for periodic replies in demos does not work yet
- **0.0.0.8:** 2008-05-26
 - added compatibility code for MS Visual C++ Compiler (VCC)
 - * sdhlibrary_defines with compatibility macros
 - * pragmas for VCC
 - * _attribute_ are switched off for VCC

- * all SDH specific classes can be put in a namespace called "SDH"
 - added index-overview.html with overview of distributed files
 - index.html files in distribution are parsed for \${PROJECT_*}
- **0.0.0.7-b:** 2008-05-21
 - CAN timeout is now correctly set
 - fixed bug in cpp/Makefile: OSNAME_LINUX=1 was always appended to EXTRACPPFLAGS no matter what OS was used
 - renamed interace member to comm_interface
- **0.0.0.7-a:** 2008-05-17
 - bug fix: minor changes to make compilation work on Linux. (ESDs ntcn.h for Windows is different from that for Linux)
 - still contains a bug that can be fixed without recompilation:
 - * The Linux version of ntcn canOpen does not accept timeout values < 0
 - * Workaround for demo programs: Use an additional "-T 0.0" command line parameter
- **0.0.0.7:** 2008-05-16
 - added C++ support for CAN using ESD cards
 - * restructured low level communication:
 - new base class cSerialBase (+new exception classes cSerialBase-Exception, cCANSerialESDException)
 - added support for variable baudrate for RS232 communication
 - made command line option handling in c++ demo programs more generic
 - corrected a bug in SDHLibrary-CPP that caused a SEGV (empty throw statement outside of a catch block)
- **0.0.0.6:** 2007-12-27
 - release for RoboCluster, Denmark
 - * added ref command and demo-ref program (needed for SDH-003)
 - * corrected minor errors to make the above work
- **0.0.0.5-a:** 2007-06-06
 - Included bugfixes from release 0.0.0.3-a (bugfix release for Uni Wales, see below) into release for care-o-bot
- **0.0.0.5:** 2007-05-24
 - Release for care-o-bot (IPA, Stuttgart), mai 2007

- Restructured files: library stuff into sdh/ and demo programs in demo/ to ease installation on user platform
 - Added library support for the new firmware features:
 - * cSDHSerial: a() vp(), vel()
 - * cSDH: Get/SetAxisAcceleration(), GetAxisMaxAcceleration(), Get/SetVelocityProfile(), GetAxisCurrentVelocity()
 - while preparing release for IPA care-o-bot:
 - * since line endings are corrected in firmware now removed the special EOL treatment in readline
 - * enhanced generation of distribution
 - * extended README files
 - * added demo-simple3 in cpp and python
 - * made compilation work on linux without warnings (SuSE 8.1 and Knoppix_v5.1.1)
 - * added requested functions GetAxisActualState() and WaitAxis() in cpp and python library
 - * added eAxisState enums from firmware
 - * corrected some yet undetected errors
 - * corrected / enhanced some doxygen comments
 - * tried to find bug:
 - firmware not moving from 5,-5,0,0,0,0,0 to 20,0,0,0,0,0,0:
 - axis 1 is stuck at 1.4...
 - bug could not be resolved (does not happen for larger movements)
- **0.0.0.4:** 2007-03-19
- Release for demo at NASA, march 2007
 - * adjusted expected lines for "m" command (it now prints one line debug output for every axis)
- **0.0.0.3-a:** 2007-06-05
- Release modified according to bug report from Martin Huelse
 - * A cSDH object could be opened successfully even if no SDH was connected or was connected but not powered:
 - added demo-test program to verify erroneous / repaired behaviour
 - added SetTimeout and GetTimeout to cRS232 class
 - made the code to verify proper connection to SDH work
 - * Exceptions could not be caught properly:
 - corrected some real printf-style format string related problems in creations of exceptions
 - added gcc style printf-style format string checking (to enable the compiler to detect errors like the above at compile time)

- The following piece of information from the C++ Annotations was not considered properly. See <http://www.icce.rug.nl/documents/cplusplus> "A function for which a function throw list is specified may not throw other types of exceptions. A run-time error occurs if it tries to throw other types of exceptions than those mentioned in the function throw list."
 - corrected the function throw lists/exception specification lists so that the most generic type thrown was listed. Thus all the user-level functions now just mention `cSDHLibraryException*` in their function throw list as all thrown exceptions are derived from it.
- Further changes
 - * added further function throw lists/exception specification lists
 - * merged in some changes from newer releases (up to 0.0.0.5):
 - retrying of sending in case of transmission errors
 - naming of sequential / non sequential (formerly called synchronous/asynchronous)
 - fixed many typos in doxygen comments
- **0.0.0.3:** 2007-03-09
 - Release modified at visit Uni-Wales
 - * Changes to make everything work on Ubuntu-Linux
 - * Enhanced Makefile a little bit to be more comfortable for the end user
- **0.0.0.2:** 2007-03-07
 - release, for Uni-Wales
- **0.0.0.1**
 - initial release, works for the first time, but not reliably

11.54 sdh/rs232-cygwin.cpp File Reference

Implementation of class `SDH::cRS232`, a class to access serial RS232 port on cygwin/linux.

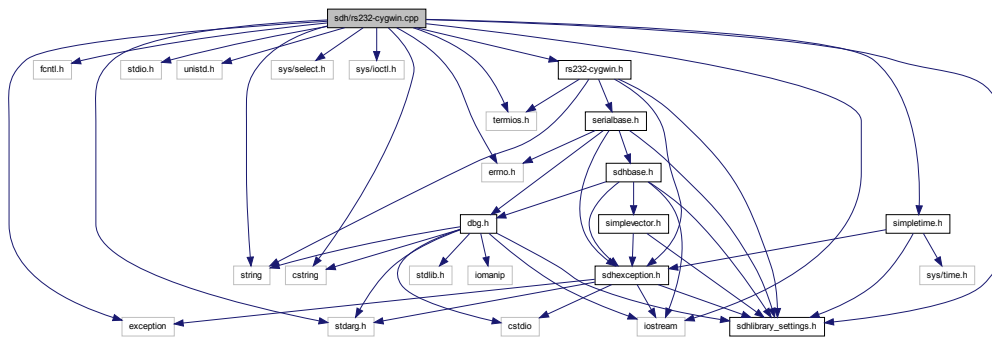
```
#include "sdhlibrary_settings.h"
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <sys/select.h>
```

```

#include <sys/ioctl.h>
#include <iostream>
#include <exception>
#include <string>
#include <stdarg.h>
#include <cstring>
#include "rs232-cygwin.h"
#include "simpletime.h"

```

Include dependency graph for rs232-cygwin.cpp:



Defines

- #define [SDH_RS232_CYGWIN_DEBUG](#) 1
- #define [DBG\(...\)](#)

Functions

- char * [StrDupNew](#) (char *const s)
helper function, duplicate string s into a char array allocated with new[]

11.54.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

11.54.2 General file information

Author

Dirk Osswald

Date

2007-02-20

11.54.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.54.4 Define Documentation

11.54.4.1 #define DBG(...)

Value:

```
do {
    __VA_ARGS__;
} while (0)
```

instead of guarding every debug output with `#if SDH_RS232_CYGWIN_DEBUG / #endif` we use this `DBG` macro that expands to a stream output to a `dbg` object or to `"`; depending on the value of `SDH_RS232_CYGWIN_DEBUG`

11.54.4.2 #define SDH_RS232_CYGWIN_DEBUG 1

Flag, if true then code for debug messages is included.

The debug messages must still be enabled at run time by setting the `some_crs232_object.dbg.SetFlag(1)`.

This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

11.54.5 Function Documentation

11.54.5.1 char* StrDupNew (char *const s)

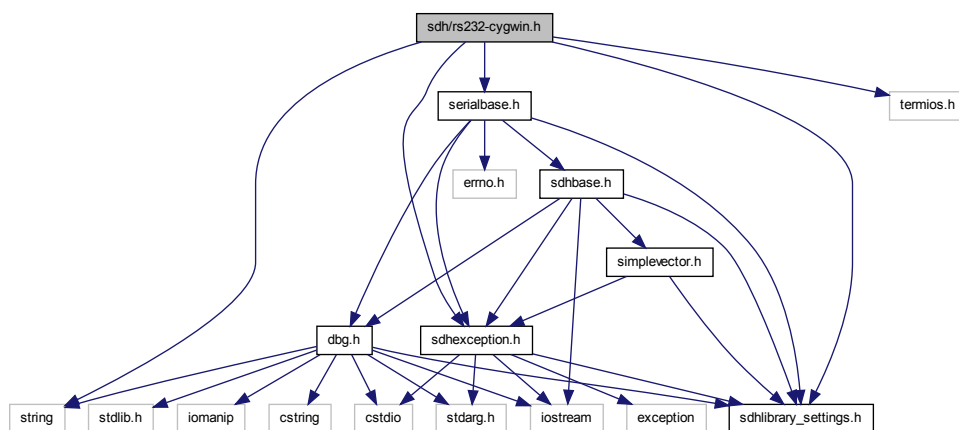
helper function, duplicate string `s` into a char array allocated with `new[]`

11.55 sdh/rs232-cygwin.h File Reference

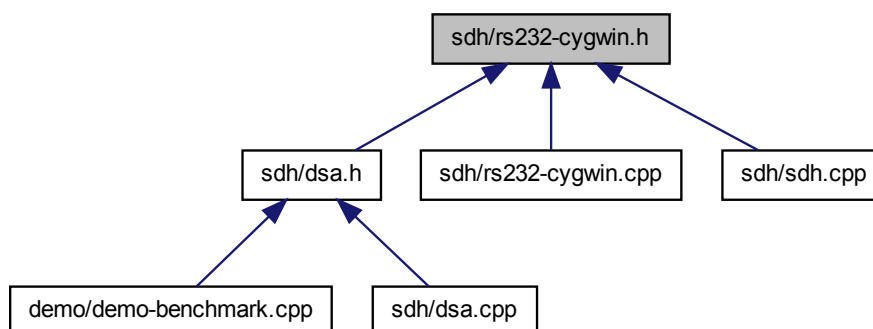
Interface of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

```
#include <string>
#include <termios.h>
#include "sdhexception.h"
#include "serialbase.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for rs232-cygwin.h:



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



Classes

- class [SDH::cRS232Exception](#)
Derived exception class for low-level RS232 related exceptions.
- class [SDH::cRS232](#)
Low-level communication class to access a serial port on Cygwin and Linux.

Namespaces

- namespace [SDH](#)

Variables

- [SDH::cRS232Exception SDH::SDH__attribute__](#)

11.55.1 Detailed Description

Interface of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

11.55.2 General file information

Author

Dirk Osswald

Date

2007-02-20

11.55.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

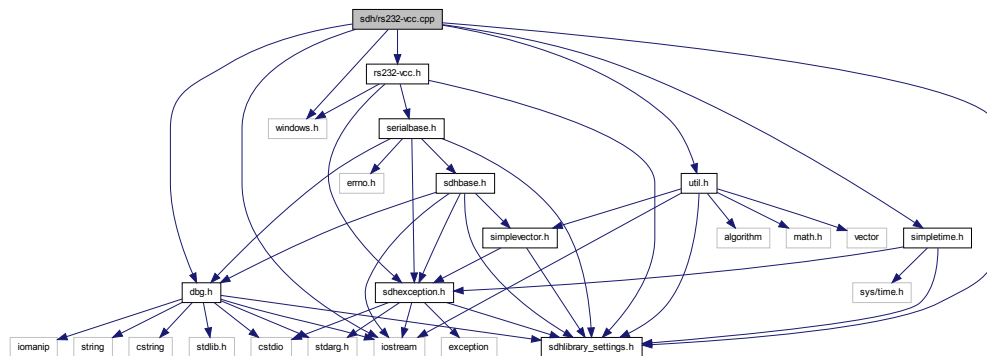
11.56 sdh/rs232-vcc.cpp File Reference

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

```
#include "iostream"
#include <windows.h>
#include "rs232-vcc.h"
#include "simpletime.h"
```

```
#include "sdhlibrary_settings.h"
#include "util.h"
#include "dbg.h"
```

Include dependency graph for rs232-vcc.cpp:



Defines

- `#define _CRT_SECURE_NO_WARNINGS 1`
- `#define SDH_RS232_VCC_DEBUG 1`
- `#define DBG(...)`

11.56.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

11.56.2 General file information

Author

Martin

Date

2008-05-23

11.56.3 Copyright

Code kindly provided by Martin from the RoboCluster project Denmark.

11.56.4 Define Documentation

11.56.4.1 `#define _CRT_SECURE_NO_WARNINGS 1`

11.56.4.2 `#define DBG(...)`

Value:

```
do {  
    __VA_ARGS__;  
} while (0)
```

11.56.4.3 `#define SDH_RS232_VCC_DEBUG 1`

Flag, if true then code for debug messages is included.

The debug messages must still be enabled at run time by setting the `some_cRS232_object.dbg.SetFlag(1)`.

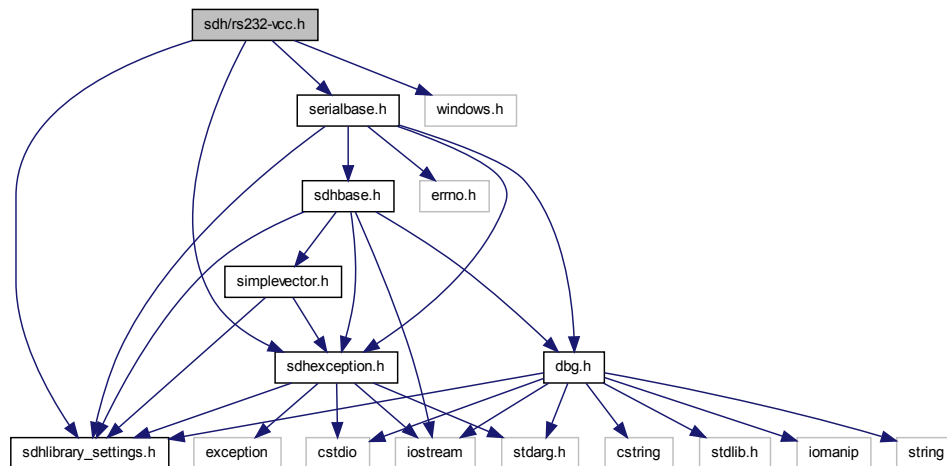
This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

11.57 sdh/rs232-vcc.h File Reference

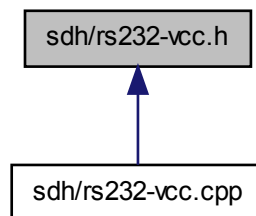
Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

```
#include "sdhlibrary_settings.h"  
#include <windows.h>  
#include "sdhexception.h"  
#include "serialbase.h"
```


Include dependency graph for rs232-vcc.h:



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



Classes

- class [SDH::cRS232Exception](#)
Derived exception class for low-level RS232 related exceptions.
- class [SDH::cRS232](#)
Low-level communication class to access a serial port on Cygwin and Linux.

Namespaces

- namespace [SDH](#)

Defines

- `#define` [SDH_RS232_VCC_ASYNC](#) 0

11.57.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

11.57.2 General file information

Author

Martin

Date

2008-05-23

11.57.3 Copyright

Code kindly provided by Martin from the RoboCluster project Denmark.

11.57.4 Define Documentation

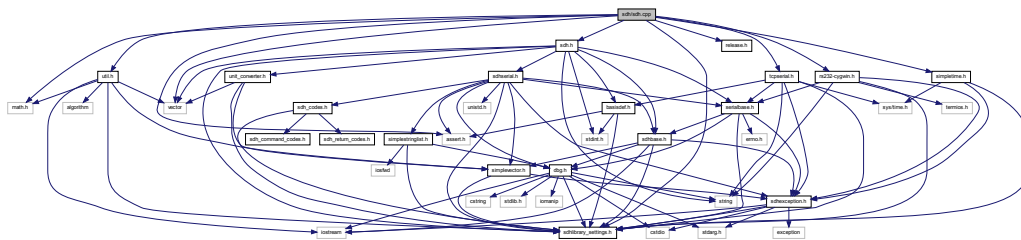
11.57.4.1 `#define` [SDH_RS232_VCC_ASYNC](#) 0

11.58 sdh/sdh.cpp File Reference

This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

```
#include <math.h>
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <vector>
#include "sdh.h"
#include "util.h"
#include "release.h"
```

Include dependency graph for sdh.cpp:



- `#define _USE_MATH_DEFINES`

This file contains the interface to class `SDH::cSDH`, the end user class to access the SDH from a PC.

Author

Date

11.58.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.58.4 Define Documentation

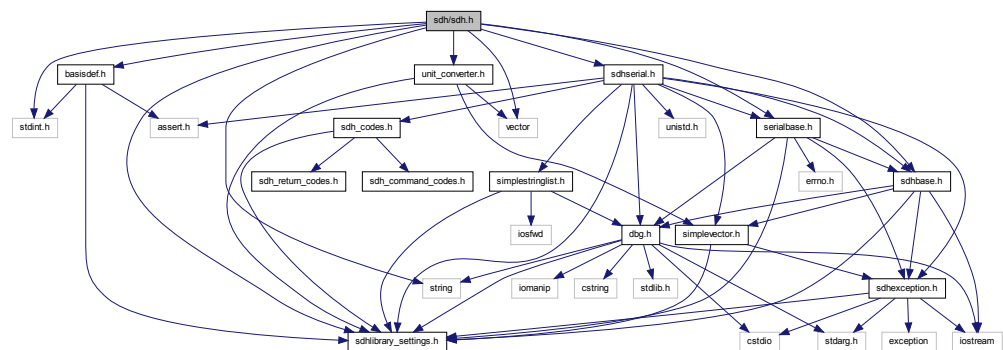
11.58.4.1 #define USE_MATH_DEFINES

11.59 sdh/sdh.h File Reference

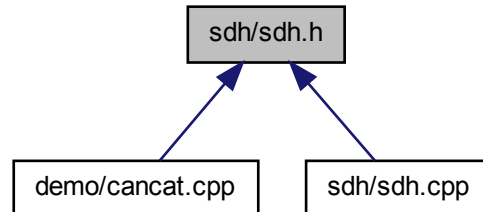
This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

```
#include "sdhlibrary_settings.h"
#include <stdint.h>
#include "basisdef.h"
#include <vector>
#include <string>
#include "sdhbase.h"
#include "sdhserial.h"
#include "unit_converter.h"
#include "serialbase.h"
```

Include dependency graph for sdh.h:



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



Classes

- class [SDH::cSDH](#)
SDH::cSDH is the end user interface class to control a SDH (SCHUNK Dexterous Hand).

Namespaces

- namespace [SDH](#)

11.59.1 Detailed Description

This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

11.59.2 General file information

Author

Dirk Osswald

Date

2007-02-20

11.59.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

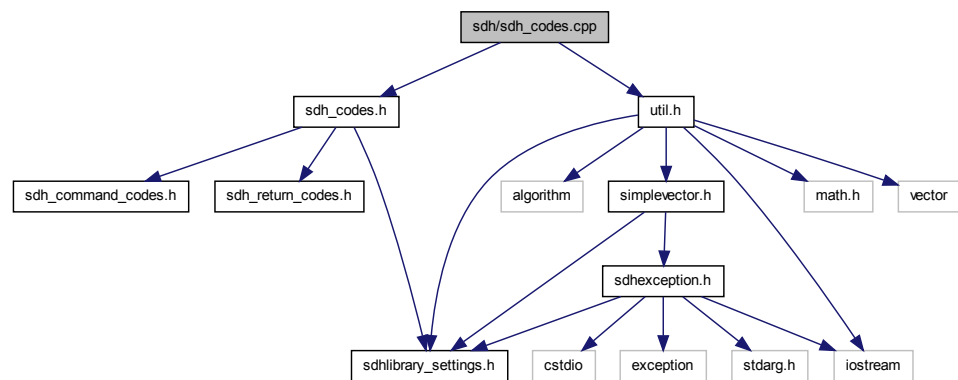
11.60 sdh/sdh_codes.cpp File Reference

This file contains function to convert the binary command and return codes of the [SDH](#) to strings.

```
#include "sdh_codes.h"
```

```
#include "util.h"
```

Include dependency graph for sdh_codes.cpp:



11.60.1 Detailed Description

This file contains function to convert the binary command and return codes of the [SDH](#) to strings.

11.60.2 General file information

Author

Dirk Osswald

Date

2011-02-04

11.60.3 Copyright

- Copyright (c) 2011 SCHUNK GmbH & Co. KG

11.61 sdh/sdh_codes.h File Reference

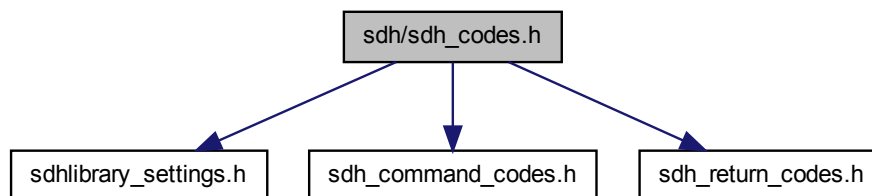
This file contains function to convert the binary command codes of the [SDH](#). To use this from a non gcc compiler you might have to define `SDH__attribute__` to nothing and `SDH_USE_VCC` to 1.

```
#include "sdhlibrary_settings.h"
```

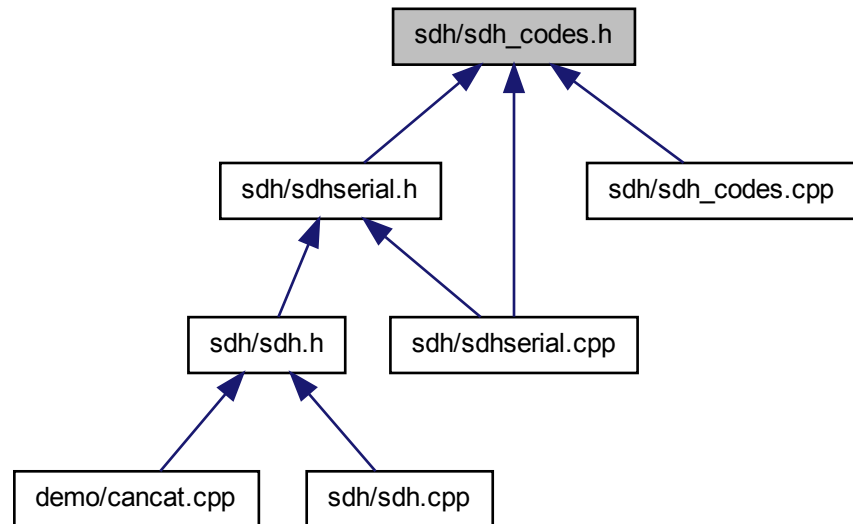
```
#include "sdh_command_codes.h"
```

```
#include "sdh_return_codes.h"
```

Include dependency graph for `sdh_codes.h`:



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



Namespaces

- namespace [SDH](#)

Functions

- char const * [SDH::SDHCommandCodeToString](#) (eCommandCode cc)
- char const * [SDH::SDHReturnCodeToString](#) (eReturnCode rc)

11.61.1 Detailed Description

This file contains function to convert the binary command codes of the [SDH](#). To use this from a non gcc compiler you might have to define `SDH__attribute__` to nothing and `SDH_USE_VCC` to 1.

11.61.2 General file information

Author

Dirk Osswald

Date

2011-02-04

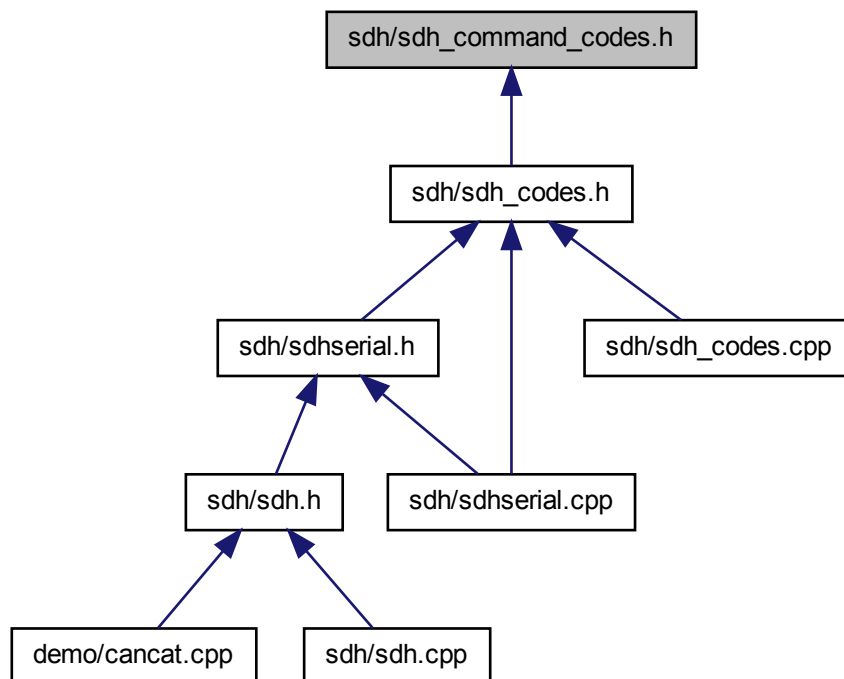
11.61.3 Copyright

- Copyright (c) 2003 IPR, University of Karlsruhe (TH), all rights reserved
- Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.62 sdh/sdh_command_codes.h File Reference

This file contains the binary command codes of the [SDH](#). To use this from a non gcc compiler you might have to define `SDH__attribute__` to nothing and `SDH_USE_VCC` to 1.

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



Defines

- `#define USE_CMD_TEST 0`
- `#define SDH__attribute__(...) __attribute__((__VA_ARGS__))`
- `#define SDH_USE_VCC 0`

Typedefs

- `typedef enum eCommandCodeEnum eCommandCode`
typedef for eCommandCodeEnum, see there

Enumerations

- `enum eCommandCodeEnum`

Functions

- `enum eCommandCodeEnum __attribute__((packed))`

Variables

- `CMDC_V = 128`
- `CMDC_VEL`
- `CMDC_RVEL`
- `CMDC_POS`
- `CMDC_STATE`
- `CMDC_P`
- `CMDC_A`
- `CMDC_M`
- `CMDC_STOP`
- `CMDC_VP`
- `CMDC_CON`
- `CMDC_TPAP`
- `CMDC_TVAV`
- `CMDC_VLIM`
- `CMDC ALIM`
- `CMDC_POS_SAVE`
- `CMDC_REF`
- `CMDC_TEMP`
- `CMDC_ID`
- `CMDC_SN`
- `CMDC_VER`
- `CMDC_VER_DATE`
- `CMDC_SOC`

- [CMDC_SOC_DATE](#)
- [CMDC_NUMAXIS](#)
- [CMDC_P_MIN](#)
- [CMDC_P_MAX](#)
- [CMDC_P_OFFSET](#)
- [CMDC_GET_DURATION](#)
- [CMDC_IGRIP](#)
- [CMDC_IHOLD](#)
- [CMDC_SELGRIP](#)
- [CMDC_GRIP](#)
- [CMDC_PID](#)
- [CMDC_KV](#)
- [CMDC_ILIM](#)
- [CMDC_POWER](#)
- [CMDC_DEMO](#)
- [CMDC_USER_ERRORS](#)
- [CMDC_TERMINAL](#)
- [CMDC_DEBUG](#)
- [CMDC_USE_FIXED_LENGTH](#)
- [CMDC_CHANGE_RS232](#)
- [CMDC_CHANGE_CHANNEL](#)

11.62.1 Detailed Description

This file contains the binary command codes of the [SDH](#). To use this from a non gcc compiler you might have to define `SDH__attribute__` to nothing and `SDH_USE_VCC` to 1.

11.62.2 General file information

Author

Ilshat Mamaev, Dirk Osswald

Date

2011-02-01

11.62.3 Copyright

- Copyright (c) 2003 IPR, University of Karlsruhe (TH), all rights reserved
- Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.62.4 Define Documentation

11.62.4.1 `#define SDH__attribute__(...) __attribute__((VA_ARGS__))`

11.62.4.2 `#define SDH_USE_VCC 0`

11.62.4.3 `#define USE_CMD_TEST 0`

11.62.5 Typedef Documentation

11.62.5.1 `typedef enum eCommandCodeEnum eCommandCode`

typedef for eCommandCodeEnum, see there

11.62.6 Enumeration Type Documentation

11.62.6.1 `enum eCommandCodeEnum`

Packed (1 Byte) enum with binary command codes used to indicate the command sent in the binary communication request or response.

Remarks

- Not all commands are implemented as binary commands
- To make this definition work with both C (nios-gcc) and C++ (std gcc/VCC) we have to separate the enum definition from the typedef.
- You may use the corresponding typedef eCommandCode

11.62.7 Function Documentation

11.62.7.1 `enum eCommandCodeEnum __attribute__((packed))`

11.62.8 Variable Documentation

11.62.8.1 `CMDC_A`

11.62.8.2 `CMDC ALIM`

11.62.8.3 `CMDC_CHANGE_CHANNEL`

11.62.8.4 `CMDC_CHANGE_RS232`

11.62.8.5 `CMDC_CON`

11.62.8.6 `CMDC_DEBUG`

11.62.8.7 `CMDC_DEMO`

11.62.8.8 `CMDC_GET_DURATION`

11.62.8.9 `CMDC_GRIP`

11.62.8.10 `CMDC_ID`

11.62.8.11 `CMDC_IGRIP`

11.62.8.12 `CMDC_IHOLD`

11.62.8.13 `CMDC_ILIM`

11.62.8.14 `CMDC_KV`

11.62.8.15 `CMDC_M`

11.62.8.16 `CMDC_NUMAXIS`

11.62.8.17 `CMDC_P`

11.62.8.18 `CMDC_P_MAX`

11.62.8.19 `CMDC_P_MIN`

11.62.8.20 `CMDC_P_OFFSET`

11.62.8.21 `CMDC_PID`

11.62.8.22 `CMDC_POS`

11.62.8.23 `CMDC_POS_SAVE`

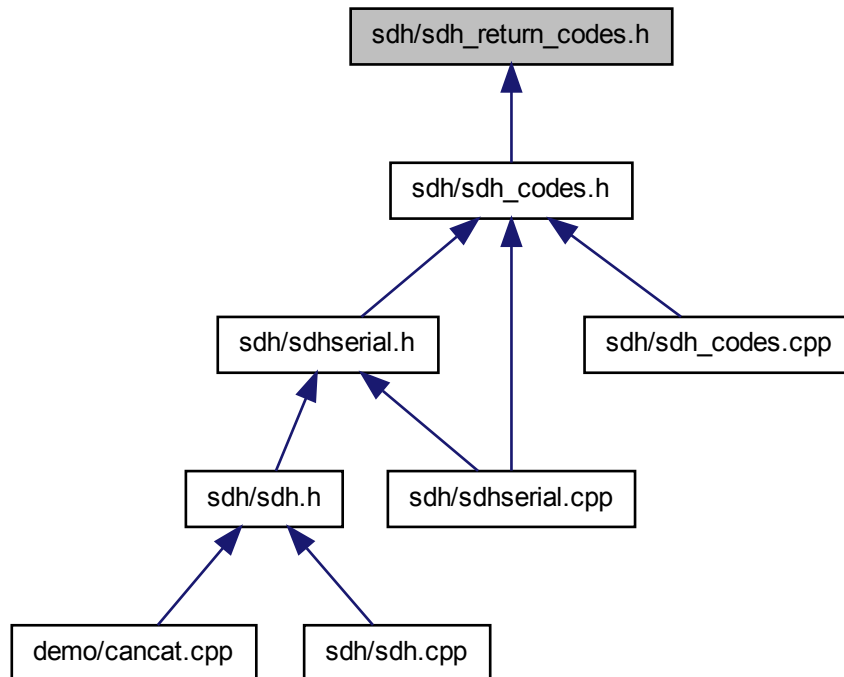
11.62.8.24 `CMDC_POWER`

11.62.8.25 `CMDC_REF`

11.62.8.26 `CMDC_RVEL`

11.62.8.27 `CMDC_SELGRIP`

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



Defines

- #define [SDH__attribute__](#)(...) [__attribute__\(\(__VA_ARGS__\)\)](#)
- #define [SDH_USE_VCC](#) 0

Typedefs

- typedef enum [eReturnCodeEnum](#) [eReturnCode](#)
typedef for eCommandCodeEnum, see there

Enumerations

- enum [eReturnCodeEnum](#)

Functions

- enum [eReturnCodeEnum](#) [SDH__attribute__ \(\(__packed__\)\)](#)

Variables

- [RC_OK](#) = 0
Success, no error.
- [RC_NOT_AVAILABLE](#)
Error: An accessed ressource is not available.
- [RC_NOT_INITIALIZED](#)
Error: An accessed ressource has not been initialized.
- [RC_ALREADY_RUNNING](#)
Error: Data acquisition: the acquisition loop is already running.
- [RC_FEATURE_NOT_SUPPORTED](#)
- [RC_INCONSISTENT_DATA](#)
- [RC_TIMEOUT](#)
Error: timeout occured.
- [RC_READ_ERROR](#)
Error: could not read.
- [RC_WRITE_ERROR](#)
Error: could not write.
- [RC_INSUFFICIENT_RESOURCES](#)
Error: Insufficient ressources.
- [RC_CHECKSUM_ERROR](#)
- [RC_NOT_ENOUGH_PARAMS](#)
Error: not enough parameters on command line.
- [RC_NO_PARAMS_EXPECTED](#)
- [RC_CMD_UNKNOWN](#)
Error: unknown command on command line.
- [RC_CMD_FORMAT_ERROR](#)
Error: invalid format of command line parameters.
- [RC_ACCESS_DENIED](#)
- [RC_ALREADY_OPEN](#)
- [RC_CMD_FAILED](#)

- [RC_CMD_ABORTED](#)
- [RC_INVALID_HANDLE](#)
- [RC_DEVICE_NOT_FOUND](#)
- [RC_DEVICE_NOT_OPENED](#)
- [RC_IO_ERROR](#)

Error: Input/Output error like bus-off detected.

- [RC_INVALID_PARAMETER](#)

Error: invalid parameter on command line.

- [RC_RANGE_ERROR](#)
- [RC_NO_DATAPIPE](#)
- [RC_INDEX_OUT_OF_BOUNDS](#)

Error: A given index parameter is invalid.

- [RC_HOMING_ERROR](#)
- [RC_AXIS_DISABLED](#)
- [RC_OVER_TEMPERATURE](#)
- [RC_MAX_COMMANDS_EXCEEDED](#)

Error: cannot add more than CI_MAX_COMMANDS to interpreter / POSCON_MAX_OSCILLOSCOPE parameters to oscilloscope.

- [RC_INVALID_PASSWORD](#)

Error: invalid password given for change user command.

- [RC_MAX_COMMANDLINE_EXCEEDED](#)

Error: the command line given is too long.

- [RC_CRC_ERROR](#)

Cyclic Redundancy Code error while receiving binary input.

- [RC_NO_COMMAND](#)

Not really an error: reading input did not yield a new command.

- [RC_INTERNAL](#)

Error: callback function reports internal error.

- [RC_UNKNOWN_ERROR](#)

Error: unknown error.

- [RC_DIMENSION](#)

< End marker and dimension

11.63.1 Detailed Description

This file contains a typedef for a common Return Codes enum.

11.63.2 General file information

Author

Dirk Osswald

Date

2011-02-04

11.63.3 Copyright

Copyright (c) 2011 SCHUNK GmbH & Co. KG

11.63.4 Define Documentation

11.63.4.1 `#define SDH__attribute__(...) __attribute__(__VA_ARGS__)`

11.63.4.2 `#define SDH_USE_VCC 0`

11.63.5 Typedef Documentation

11.63.5.1 `typedef enum eReturnCodeEnum eReturnCode`

typedef for eCommandCodeEnum, see there

11.63.6 Enumeration Type Documentation

11.63.6.1 `enum eReturnCodeEnum`

Packed (1 Byte) enum with binary return codes used to indicate the status of the [SDH](#) sent in the binary communication request or response.

Remarks

- To make this definition work with both C (nios-gcc) and C++ (std gcc/VCC) we have to separate the enum definition from the typedef.
- You may use the corresponding typedef eReturnCode

11.63.7 Function Documentation

11.63.7.1 `enum eReturnCodeEnum SDH__attribute__((__packed__))`

11.63.8 Variable Documentation

11.63.8.1 **RC_ACCESS_DENIED**

11.63.8.2 **RC_ALREADY_OPEN**

11.63.8.3 **RC_ALREADY_RUNNING**

Error: Data acquisition: the acquisition loop is already running.

11.63.8.4 **RC_AXIS_DISABLED**

11.63.8.5 **RC_CHECKSUM_ERROR**

11.63.8.6 **RC_CMD_ABORTED**

11.63.8.7 **RC_CMD_FAILED**

11.63.8.8 **RC_CMD_FORMAT_ERROR**

Error: invalid format of command line parameters.

11.63.8.9 **RC_CMD_UNKNOWN**

Error: unknown command on command line.

11.63.8.10 **RC_CRC_ERROR**

Cyclic Redundancy Code error while receiving binary input.

11.63.8.11 **RC_DEVICE_NOT_FOUND**

11.63.8.12 **RC_DEVICE_NOT_OPENED**

11.63.8.13 **RC_DIMENSION**

< End marker and dimension

End marker and dimension.

11.63.8.14 RC_FEATURE_NOT_SUPPORTED**11.63.8.15 RC_HOMING_ERROR****11.63.8.16 RC_INCONSISTENT_DATA****11.63.8.17 RC_INDEX_OUT_OF_BOUNDS**

Error: A given index parameter is invalid.

11.63.8.18 RC_INSUFFICIENT_RESOURCES

Error: Insufficient resources.

11.63.8.19 RC_INTERNAL

Error: callback function reports internal error.

11.63.8.20 RC_INVALID_HANDLE**11.63.8.21 RC_INVALID_PARAMETER**

Error: invalid parameter on command line.

11.63.8.22 RC_INVALID_PASSWORD

Error: invalid password given for change user command.

11.63.8.23 RC_IO_ERROR

Error: Input/Output error like bus-off detected.

11.63.8.24 RC_MAX_COMMANDLINE_EXCEEDED

Error: the command line given is too long.

11.63.8.25 RC_MAX_COMMANDS_EXCEEDED

Error: cannot add more than CI_MAX_COMMANDS to interpreter / POSCON_MAX_OSCILLOSCOPE parameters to oscilloscope.

11.63.8.26 RC_NO_COMMAND

Not really an error: reading input did not yield a new command.

11.63.8.27 RC_NO_DATAPIPE**11.63.8.28 RC_NO_PARAMS_EXPECTED****11.63.8.29 RC_NOT_AVAILABLE**

Error: An accessed ressource is not available.

11.63.8.30 RC_NOT_ENOUGH_PARAMS

Error: not enough parameters on command line.

11.63.8.31 RC_NOT_INITIALIZED

Error: An accessed ressource has not been initialized.

11.63.8.32 RC_OK = 0

Success, no error.

11.63.8.33 RC_OVER_TEMPERATURE**11.63.8.34 RC_RANGE_ERROR****11.63.8.35 RC_READ_ERROR**

Error: could not read.

11.63.8.36 RC_TIMEOUT

Error: timeout occurred.

11.63.8.37 RC_UNKNOWN_ERROR

Error: unknown error.

11.63.8.38 RC_WRITE_ERROR

Error: could not write.

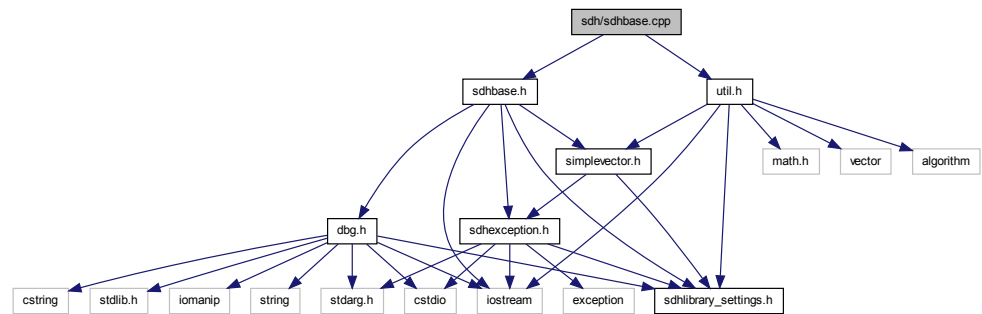
11.64 sdh/sdhbase.cpp File Reference

Implementation of class [SDH::cSDHBase](#).

```
#include "sdhbase.h"
```

```
#include "util.h"
```

Include dependency graph for `sdhbase.cpp`:



Namespaces

- namespace [SDH](#)

Variables

- `std::ostream * SDH::g_sdh_debug_log = &std::cerr`

11.64.1 Detailed Description

Implementation of class [SDH::cSDHBase](#).

11.64.2 General file information

Author

Dirk Osswald

Date

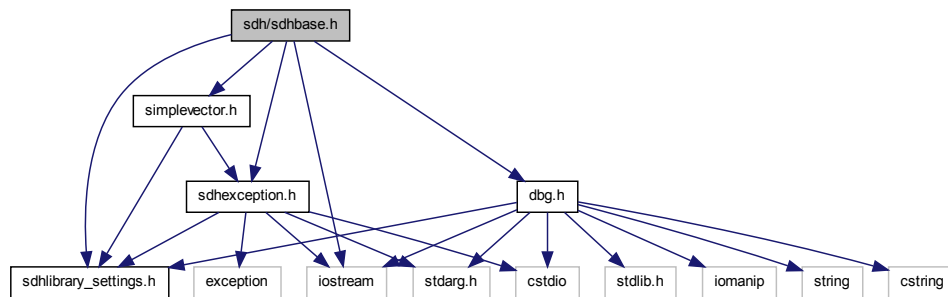
2007-02-19

11.64.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

Interface of class `SDH::cSDHBase`.

Include dependency graph for sdhbase.h:



- class [SDH::cSDHErrorInvalidParameter](#)
Derived exception class for exceptions related to invalid parameters.
- class [SDH::cSDHBase](#)

The base class to control the SCHUNK Dexterous Hand.

Namespaces

- namespace [SDH](#)

11.65.1 Detailed Description

Interface of class [SDH::cSDHBase](#).

11.65.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.65.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.66 sdh/sdhexception.cpp File Reference

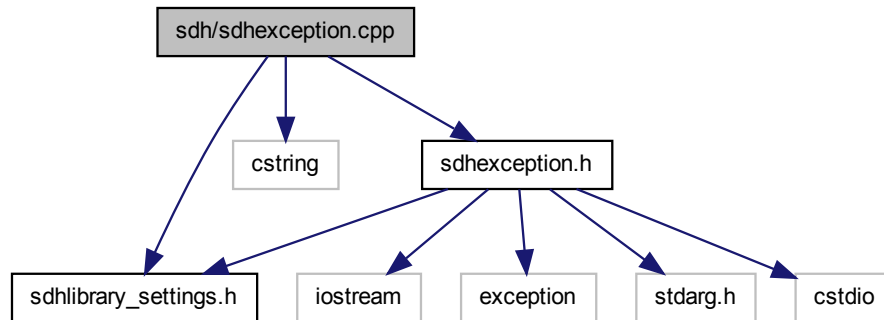
Implementation of the exception base class [SDH::cSDHLibraryException](#) and [SDH::cMsg](#).

```
#include "sdhlibrary_settings.h"
```

```
#include <cstring>
```

```
#include "sdhexception.h"
```


Include dependency graph for sdhexception.cpp:



Namespaces

- namespace [SDH](#)

Functions

- `std::ostream & SDH::operator<< (std::ostream &stream, cMsg const &msg)`
- `std::ostream & SDH::operator<< (std::ostream &stream, cSDHLibraryException const &e)`

11.66.1 Detailed Description

Implementation of the exception base class [SDH::cSDHLibraryException](#) and [SDH::cMsg](#).

11.66.2 General file information

Author

Dirk Osswald

Date

2007-02-22

11.66.3 Copyright

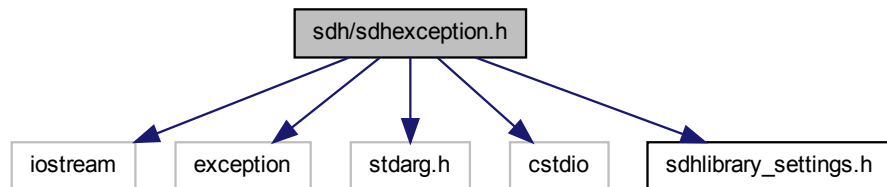
Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.67 sdh/sdhexception.h File Reference

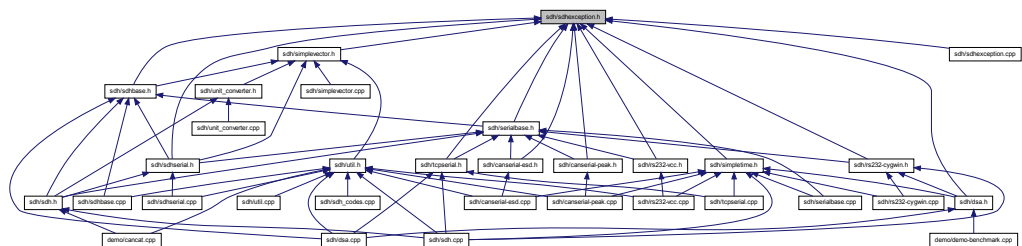
Interface of the exception base class `SDH::cSDHLibraryException` and `SDH::cMsg`.

```
#include <iostream>
#include <exception>
#include <stdarg.h>
#include <cstdio>
#include "sdhlibrary_settings.h"
```

Include dependency graph for `sdhexception.h`:



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



Classes

- class **SDH::cMsg**
Class for short, fixed maximum length text messages.
- class **SDH::cSDHLibraryException**
Base class for exceptions in the SDHLibrary-CPP.

- class [SDH::cSDHErrorCommunication](#)

Derived exception class for exceptions related to communication between the SDHLibrary and the SDH.

Namespaces

- namespace [SDH](#)

Functions

- `std::ostream & SDH::operator<< (std::ostream &stream, cMsg const &msg)`
- `std::ostream & SDH::operator<< (std::ostream &stream, cSDHLibraryException const &e)`

11.67.1 Detailed Description

Interface of the exception base class [SDH::cSDHLibraryException](#) and [SDH::cMsg](#).

11.67.2 General file information

Author

Dirk Osswald

Date

2007-02-22

11.67.3 Copyright

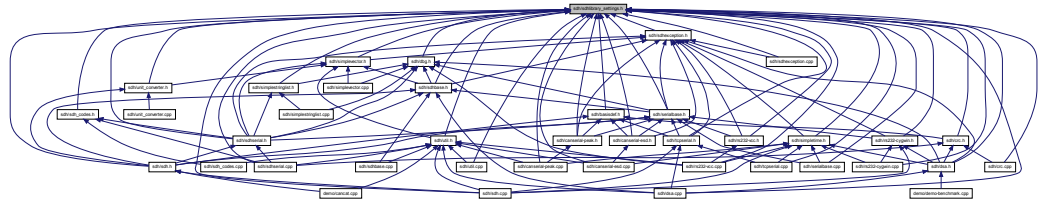
Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.68 sdh/sdhlbrary_settings.h File Reference

This file contains settings to make the SDHLibrary compile on differen systems:

- gcc/Cygwin/Windows
- gcc/Linux
- VisualC++/Windows.

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



Defines

- `#define SDH_USE_NAMESPACE 1`
Flag, if 1 then all classes are put into a namespace called `SDH`. If 0 then the classes are left outside any namespace.
- `#define SDH_USE_BINARY_COMMUNICATION 1`
Flag, if 1 then binary communication is used where possible for better performance (requires at least firmware 0.0.2.15)
- `#define NAMESPACE_SDH_START namespace SDH {`
- `#define NAMESPACE_SDH_END }`
- `#define USING_NAMESPACE_SDH using namespace SDH;`
- `#define NS_SDH SDH::`

11.68.1 Detailed Description

This file contains settings to make the SDHLibrary compile on different systems:

- gcc/Cygwin/Windows
- gcc/Linux
- VisualC++/Windows.

11.68.2 General file information

Author

Dirk Osswald

Date

2008-05-20

11.68.3 Copyright

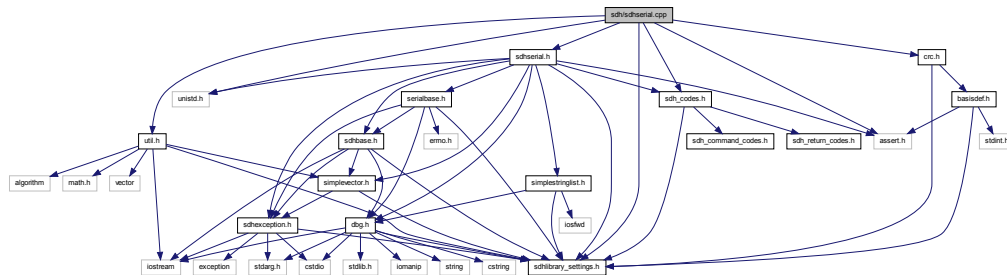
Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.69 sdh/sdhserial.cpp File Reference

Interface of class [SDH::cSDHSerial](#).

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "util.h"
#include "sdhserial.h"
#include "crc.h"
#include "sdh_codes.h"
```

Include dependency graph for sdhserial.cpp:



Classes

- struct [SDH::sSDHBinaryRequest](#)
data structure with binary data for request from PC to [SDH](#)
- struct [SDH::sSDHBinaryResponse](#)
data structure with binary data for response from [SDH](#) to PC

Namespaces

- namespace [SDH](#)

Enumerations

- enum { [SDH::eNUMBER_OF_ELEMENTS](#) = cSimpleVector::eNUMBER_OF_ELEMENTS }

Functions

- struct `SDH::sSDHBinaryRequest` `SDH::__attribute__` ((packed))
- `sSDHBinaryRequest` (`eCommandCode` command, double *value, bool use_crc16)
- `tCRCValue * CRC16` () const
return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)
- int `GetNbBytesToSend` () const
return the total number of bytes to send
- void `CheckCRC16` () const throw (cSDHErrorCommunication*)
check the CRC value in parameter_bytes. Throw an exception if check fails
- std::ostream & `SDH::operator<<` (std::ostream &stream, `sSDHBinaryRequest` const &request)
helper functions to insert a human readable form of the request into stream
- std::ostream & `SDH::operator<<` (std::ostream &stream, `sSDHBinaryResponse` const &response)
helper functions to insert a human readable form of the restore into stream

Variables

- unsigned char `cmd_code`
- unsigned char `nb_data_bytes`
- unsigned char `nb_valid_parameters`
- union {
 float `parameter` [eNUMBER_OF_ELEMENTS]
 unsigned char `parameter_bytes` [sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]
 };
- unsigned char `status_code`
- union {
 float `parameter` [eNUMBER_OF_ELEMENTS]
 unsigned char `parameter_bytes` [sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]
 };

11.69.1 Detailed Description

Interface of class `SDH::cSDHSerial`.

11.69.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.69.3 Copyright

- Copyright (c) 2014 SCHUNK GmbH & Co. KG

11.69.4 Function Documentation

11.69.4.1 void __attribute__((__nothrow__)):CheckCRC16 () const throw (cSDHErrorCommunication*)

check the CRC value in parameter_bytes. Throw an exception if check fails

11.69.4.2 tCRCValue * CRC16 () const

return a ptr to the CRC value in parameter_bytes, assuming that nb_data_bytes is correct (including the CRC bytes)

11.69.4.3 int __attribute__((__nothrow__)):GetNbBytesToSend () const

return the total number of bytes to send

11.69.4.4 __attribute__((__nothrow__)):sSDHBinaryRequest (eCommandCode *command*, double * *value*, bool *use_crc16*)

ctor, create a request with cmd_code *command* and eNUMBER_OF_ELEMENTS parameter from *value* or no parameters if *value* is NULL. Add crc if *use_crc16* is true and set nb_data_bytes appropriately

11.69.5 Variable Documentation

11.69.5.1 `union { ... }`

11.69.5.2 `union { ... }`

11.69.5.3 `unsigned char cmd_code`

11.69.5.4 `unsigned char nb_data_bytes`

11.69.5.5 `unsigned char nb_valid_parameters`

11.69.5.6 `float parameter[eNUMBER_OF_ELEMENTS]`

11.69.5.7 `unsigned char parameter_bytes[sizeof(float)*eNUMBER_OF_ELEMENTS+sizeof(tCRCValue)]`

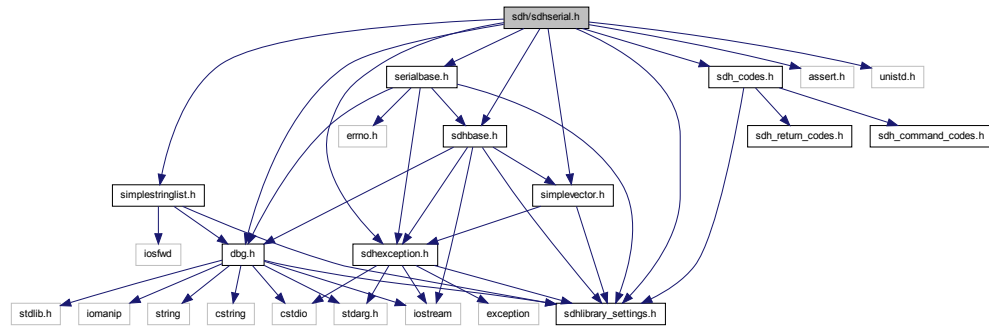
11.69.5.8 `unsigned char status_code`

11.70 sdh/sdhserial.h File Reference

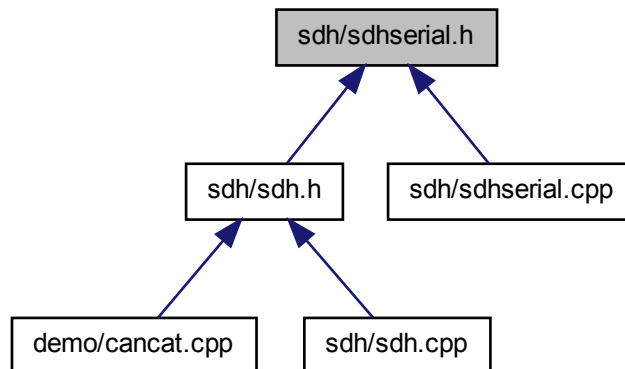
Interface of class [SDH::cSDHSerial](#).

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "dbg.h"
#include "sdhexception.h"
#include "simplevector.h"
#include "simplestringlist.h"
#include "sdhbase.h"
#include "serialbase.h"
#include "sdh_codes.h"
```


Include dependency graph for sdhserial.h:



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



Classes

- class [SDH::cSDHSerial](#)
The class to communicate with a SDH via RS232.

Namespaces

- namespace [SDH](#)

Typedefs

- typedef cSimpleVector(cSDHSerial::* [SDH::pSetFunction](#))(int, double *)
Type of a pointer to a "set-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).
- typedef cSimpleVector(cSDHSerial::* [SDH::pGetFunction](#))(int, double *)
Type of a pointer to a "get-axis-values" function like [cSDHSerial::p](#), [cSDHSerial::pos](#), ..., [cSDHSerial::igrip](#), [cSDHSerial::ihold](#) or [cSDHSerial::ilim](#).

11.70.1 Detailed Description

Interface of class [SDH::cSDHSerial](#).

11.70.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.70.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

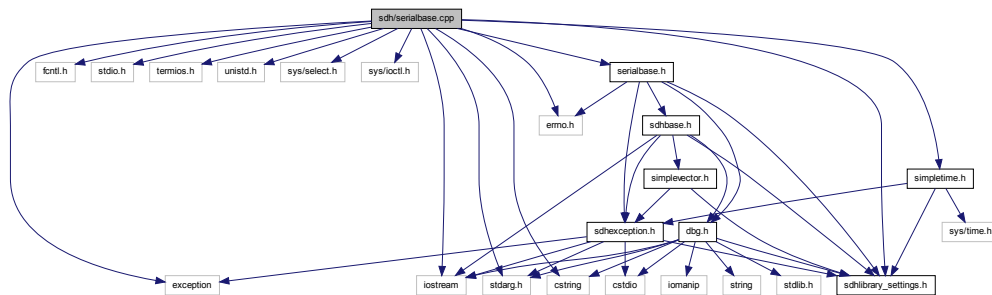
11.71 sdh/serialbase.cpp File Reference

Implementation of class [SDH::cSerialBase](#), a virtual base class to access serial interfaces like RS232 or CAN.

```
#include "sdhlibrary_settings.h"
#include <fcntl.h>
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <sys/select.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <iostream>
```

```
#include <exception>
#include <stdarg.h>
#include <cstring>
#include "serialbase.h"
#include "simpletime.h"
```

Include dependency graph for serialbase.cpp:



11.71.1 Detailed Description

Implementation of class [SDH::cSerialBase](#), a virtual base class to access serial interfaces like RS232 or CAN.

11.71.2 General file information

Author

Dirk Osswald

Date

2007-02-20

11.71.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

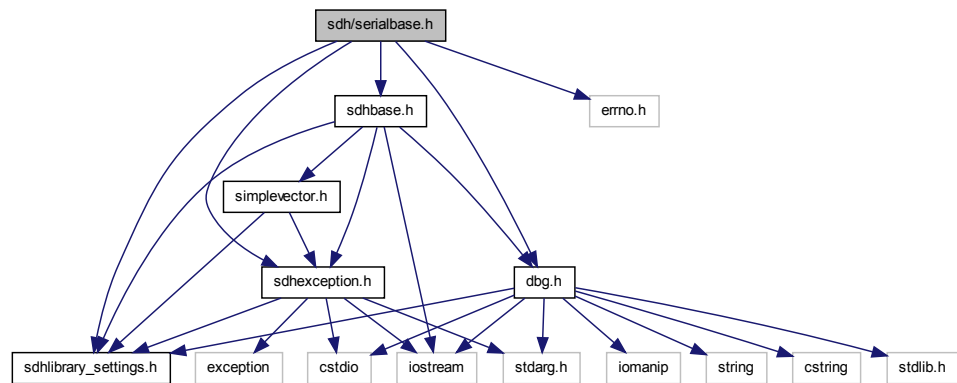
11.72 sdh/serialbase.h File Reference

Interface of class [SDH::cSerialBase](#), a virtual base class to access serial communication channels like RS232 or CAN.

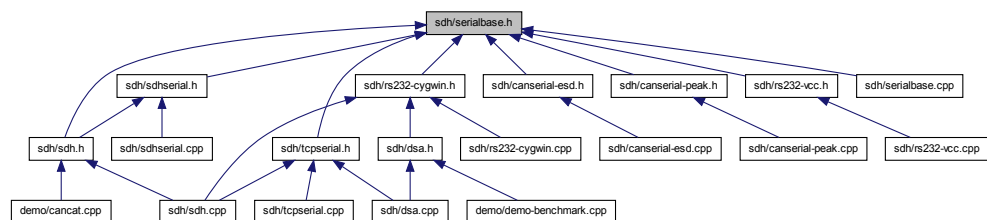
```
#include "sdhlibrary_settings.h"
```

```
#include <errno.h>
#include "sdhexception.h"
#include "dbg.h"
#include "sdhbase.h"
```

Include dependency graph for serialbase.h:



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



Classes

- class [SDH::cSerialBaseException](#)
Derived exception class for low-level serial communication related exceptions.
- class [SDH::cSerialBase](#)
Low-level communication class to access a serial port.
- class [SDH::cSerialBase::cSetTimeoutTemporarily](#)

helper class to set timeout of `_serial_base` on construction and reset to previous value on destruction. (RAII-idiom)

Namespaces

- namespace [SDH](#)

Typedefs

- typedef void * [SDH::tDeviceHandle](#)
generic device handle for CAN devices

11.72.1 Detailed Description

Interface of class [SDH::cSerialBase](#), a virtual base class to access serial communication channels like RS232 or CAN.

11.72.2 General file information

Author

Dirk Osswald

Date

2008-05-02

11.72.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.73 sdh/simplestringlist.cpp File Reference

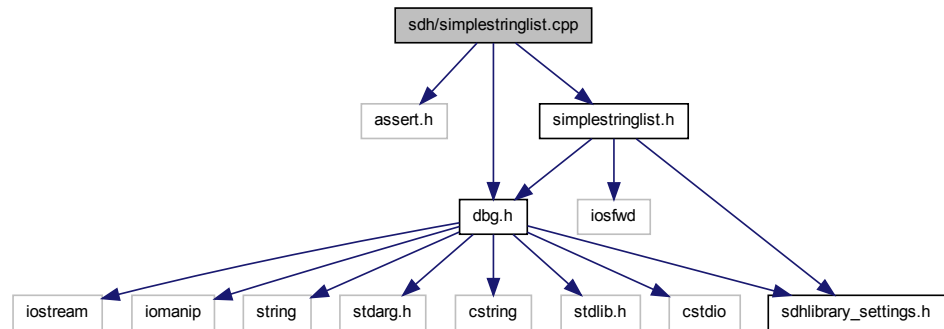
Implementation of class [SDH::cSimpleStringList](#).

```
#include <assert.h>
```

```
#include "dbg.h"
```

```
#include "simplestringlist.h"
```

Include dependency graph for simplestringlist.cpp:



Namespaces

- namespace [SDH](#)

Functions

- `std::ostream & SDH::operator<< (std::ostream &stream, cSimpleStringList const &ssl)`

Output of [cSimpleStringList](#) objects in 'normal' output streams.

11.73.1 Detailed Description

Implementation of class [SDH::cSimpleStringList](#).

11.73.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.73.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.74 sdh/simplestringlist.h File Reference

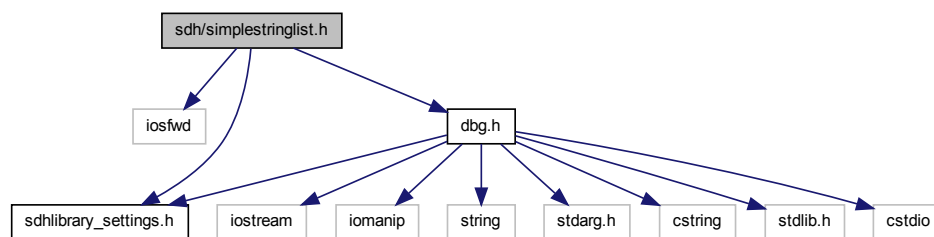
Interface of class [SDH::cSimpleStringList](#).

```
#include <iosfwd>
```

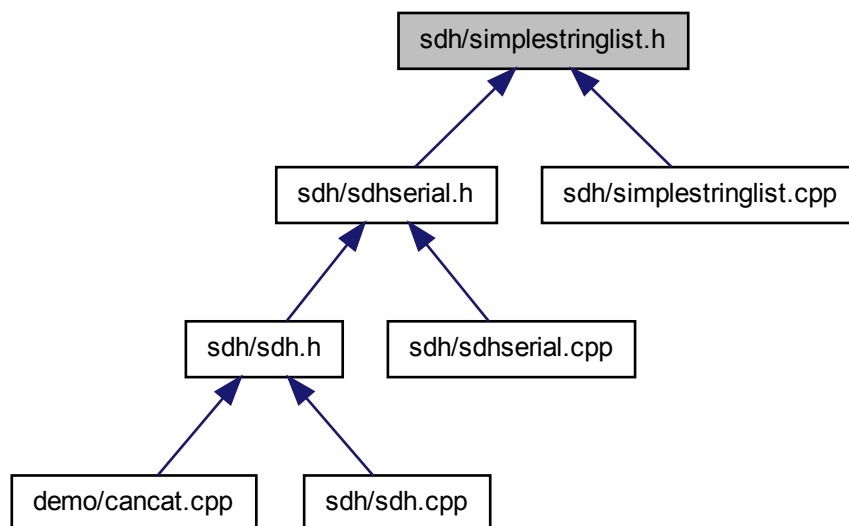
```
#include "dbg.h"
```

```
#include "sdhlibrary_settings.h"
```

Include dependency graph for simplestringlist.h:



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



Classes

- class [SDH::cSimpleStringList](#)
A simple string list. (Fixed maximum number of strings of fixed maximum length)

Namespaces

- namespace [SDH](#)

Functions

- `std::ostream & SDH::operator<< (std::ostream &stream, cSimpleStringList const &ssl)`
Output of [cSimpleStringList](#) objects in 'normal' output streams.

11.74.1 Detailed Description

Interface of class [SDH::cSimpleStringList](#).

11.74.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.74.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.75 sdh/simpletime.h File Reference

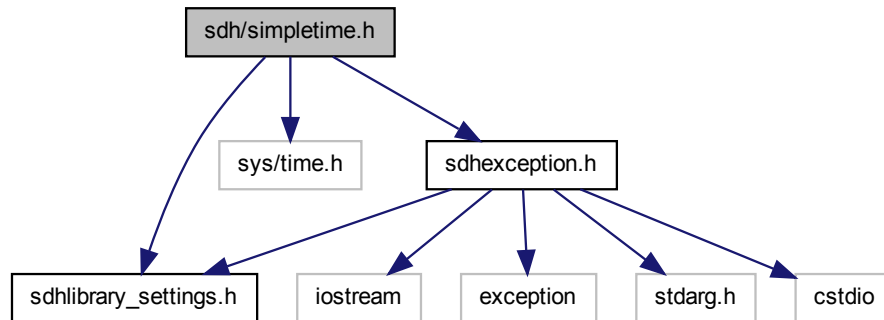
Interface of auxilliary utility functions for SDHLibrary-CPP.

```
#include "sdhlibrary_settings.h"
```

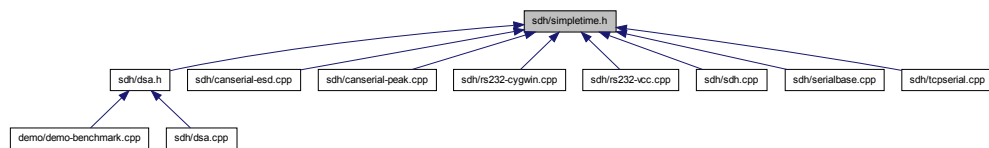
```
#include <sys/time.h>
```

```
#include "sdhexception.h"
```


Include dependency graph for simpletime.h:



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



Classes

- class [SDH::cSimpleTime](#)

Very simple class to measure elapsed time.

Namespaces

- namespace [SDH](#)

Typedefs

- typedef time_t [SDH::tTimevalSec](#)
- typedef suseconds_t [SDH::tTimevalUSec](#)

11.75.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

11.75.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.75.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.76 sdh/simplevector.cpp File Reference

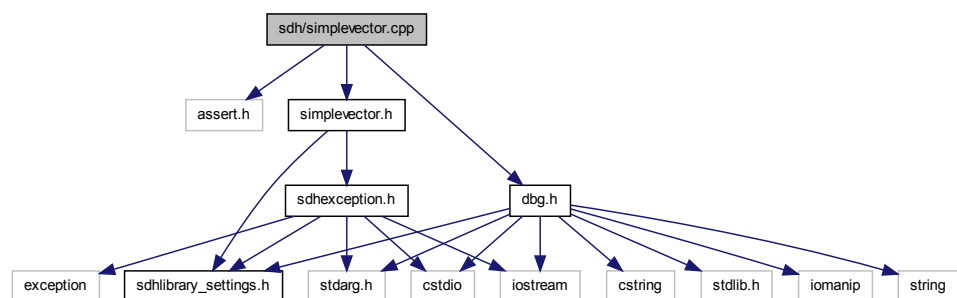
Implementation of class [SDH::cSimpleVector](#).

```
#include <assert.h>
```

```
#include "dbg.h"
```

```
#include "simplevector.h"
```

Include dependency graph for simplevector.cpp:



11.76.1 Detailed Description

Implementation of class [SDH::cSimpleVector](#).

11.76.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.76.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

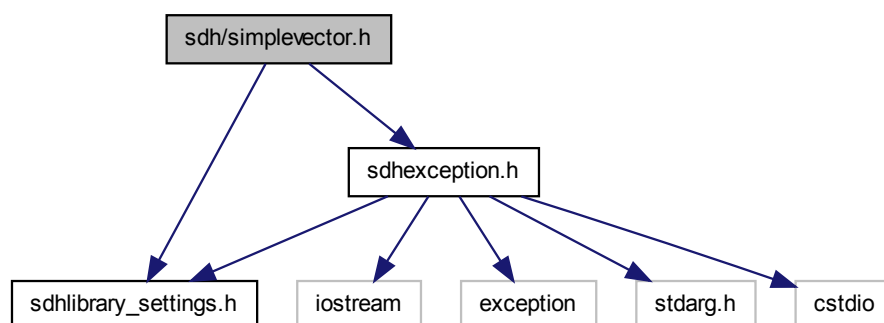
11.77 sdh/simplevector.h File Reference

Interface of class [SDH::cSimpleVector](#).

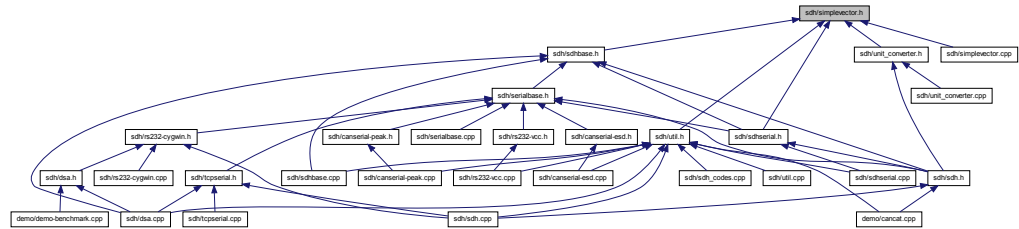
```
#include "sdhlibrary_settings.h"
```

```
#include "sdhexception.h"
```

Include dependency graph for simplevector.h:



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



Classes

- class [SDH::cSimpleVectorException](#)
Derived exception class for low-level simple vector related exceptions.
- class [SDH::cSimpleVector](#)
A simple vector implementation.

Namespaces

- namespace [SDH](#)

11.77.1 Detailed Description

Interface of class [SDH::cSimpleVector](#).

11.77.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.77.3 Copyright

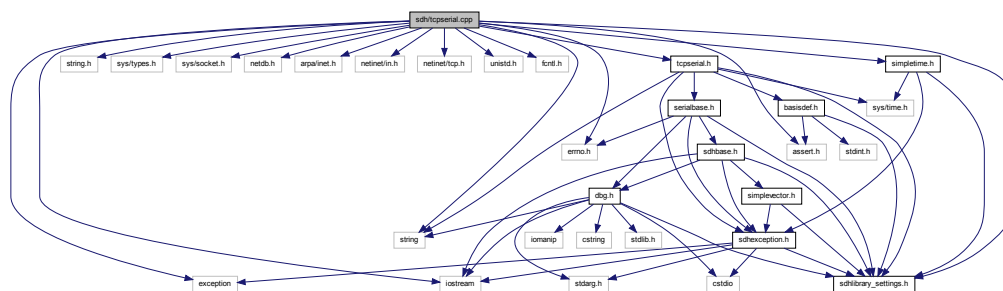
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.78 sdh/tcpserial.cpp File Reference

Implementation of class [SDH::cTCPSerial](#), a class to access a TCP port on cygwin/linux and Visual Studio.

```
#include "sdhlibrary_settings.h"
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <unistd.h>
#include <fcntl.h>
#include <iostream>
#include <exception>
#include <string>
#include <assert.h>
#include "tcpserial.h"
#include "simpletime.h"
```

Include dependency graph for tcpserial.cpp:



Defines

- #define [SDH_TCP_DEBUG](#) 1
- #define [DBG\(...\)](#)

11.78.1 Detailed Description

Implementation of class [SDH::cTCPSerial](#), a class to access a TCP port on cygwin/linux and Visual Studio.

11.78.2 General file information

Author

Dirk Osswald

Date

2010-10-30

11.78.3 Copyright

Copyright (c) 2010 SCHUNK GmbH & Co. KG

11.78.4 Define Documentation

11.78.4.1 #define DBG(...)

Value:

```
do {
    __VA_ARGS__;
} while (0)
```

instead of guarding every debug output with `#if SDH_TCP_DEBUG / #endif` we use this DBG macro that expands to a stream output to a dbg object or to ";" depending on the value of SDH_TCP_DEBUG

11.78.4.2 #define SDH_TCP_DEBUG 1

Flag, if true then code for debug messages is included.

The debug messages must still be enabled at run time by setting the `some_cTCPSerial_object.dbg.SetFlag(1)`.

This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

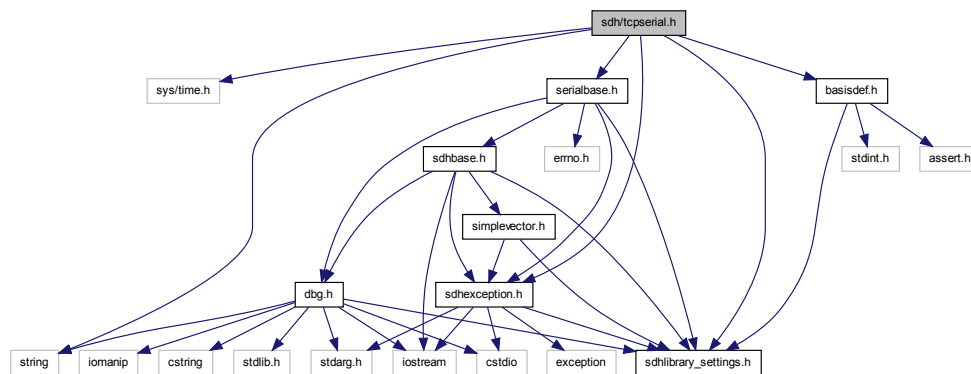
11.79 sdh/tcpserial.h File Reference

Interface of class [SDH::cTCPSerial](#), class to access TCP port cygwin/linux.

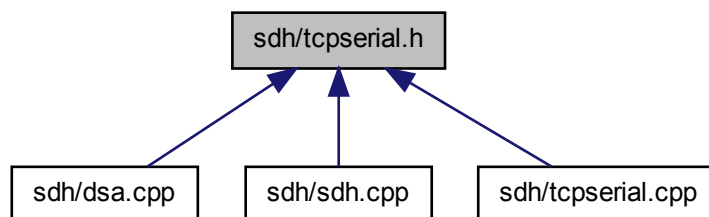
```
#include <sys/time.h>
```

```
#include <string>
#include "sdhexception.h"
#include "serialbase.h"
#include "basisdef.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for tcpserial.h:



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



Classes

- class [SDH::cTCPSerialException](#)

Derived exception class for low-level CAN ESD related exceptions.

- class [SDH::cTCPSerial](#)

Low-level communication class to access a CAN port.

Namespaces

- namespace [SDH](#)

11.79.1 Detailed Description

Interface of class [SDH::cTCPSerial](#), class to access TCP port cygwin/linux.

11.79.2 General file information

Author

Dirk Osswald

Date

2010-10-30

11.79.3 Copyright

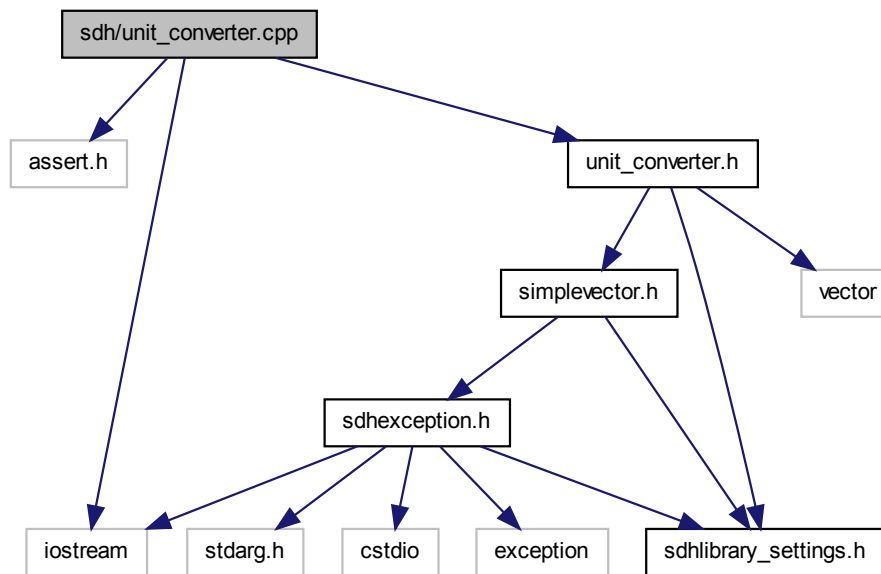
Copyright (c) 2010 SCHUNK GmbH & Co. KG

11.80 sdh/unit_converter.cpp File Reference

Implementation of class [SDH::cUnitConverter](#).

```
#include <assert.h>
#include <iostream>
#include "unit_converter.h"
```


Include dependency graph for unit_converter.cpp:



Namespaces

- namespace [SDH](#)

Variables

- `cUnitConverter` const [SDH::uc_identity](#) ("any","any","?", 1.0, 0.0, 4)
Identity converter (internal = external)

11.80.1 Detailed Description

Implementation of class [SDH::cUnitConverter](#).

11.80.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.80.3 Copyright

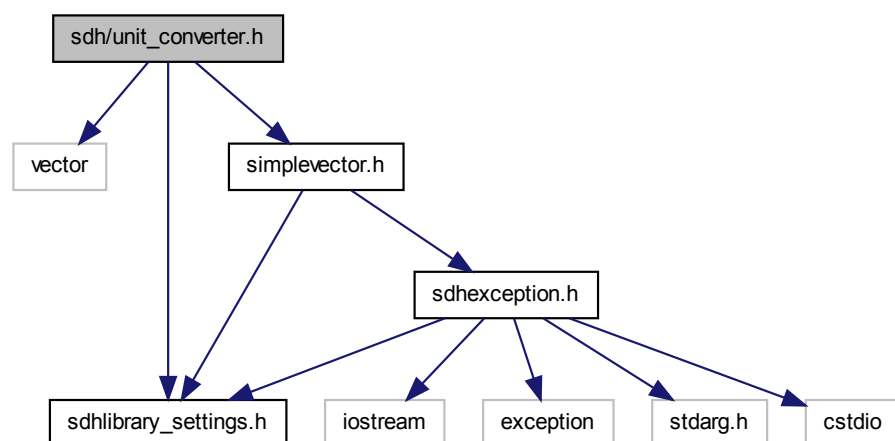
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.81 sdh/unit_converter.h File Reference

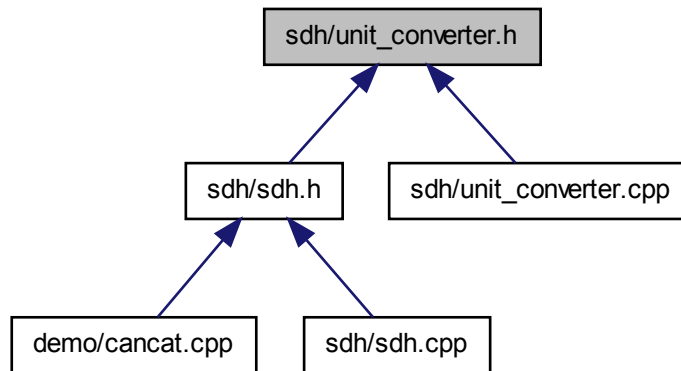
Interface of class [SDH::cUnitConverter](#).

```
#include <vector>
#include "simplevector.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for unit_converter.h:



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



Classes

- class [SDH::cUnitConverter](#)

Unit conversion class to convert values between physical unit systems.

Namespaces

- namespace [SDH](#)

Typedefs

- typedef double(cUnitConverter::* [SDH::pDoubleUnitConverterFunction](#))(double) const

Type of a pointer to a function like 'double [SDH::cUnitConverter::ToExternal\(double \) const](#)' or 'double [SDH::cUnitConverter::ToInternal\(double \) const](#)'.

11.81.1 Detailed Description

Interface of class [SDH::cUnitConverter](#).

11.81.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.81.3 Copyright

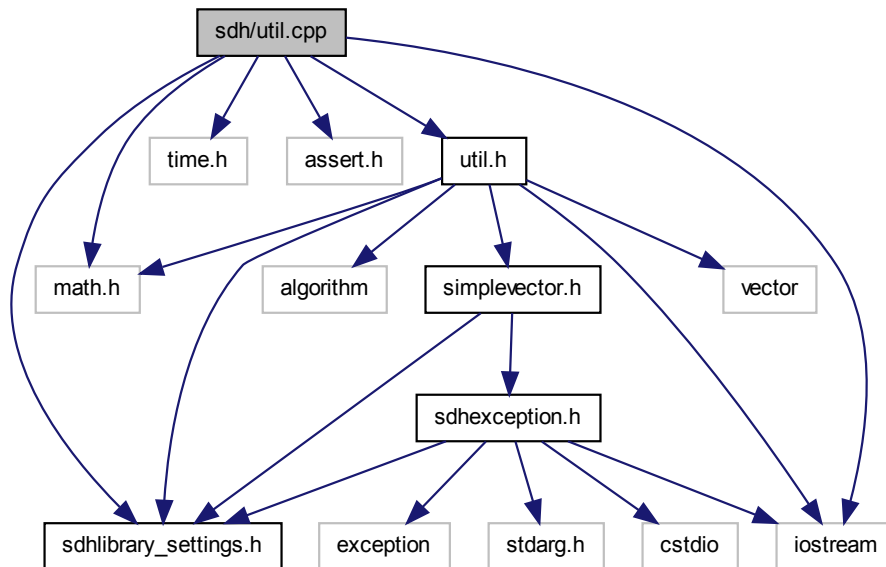
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.82 sdh/util.cpp File Reference

Implementation of auxilliary utility functions for SDHLibrary-CPP.

```
#include "sdhlibrary_settings.h"
#include <math.h>
#include <time.h>
#include <assert.h>
#include <iostream>
#include "util.h"
```

Include dependency graph for util.cpp:



Namespaces

- namespace [SDH](#)

Defines

- `#define` [_USE_MATH_DEFINES](#)

Functions

- `std::vector< int >` [SDH::NumerifyRelease](#) (char const *rev)

Auxiliary functions

- bool [SDH::InIndex](#) (int v, int max)
- bool [SDH::InRange](#) (double v, double min, double max)
- bool [SDH::InRange](#) (int n, double const *v, double const *min, double const *max)
- double [SDH::ToRange](#) (double v, double min, double max)
- void [SDH::ToRange](#) (int n, double *v, double const *min, double const *max)

- void [SDH::ToRange](#) (std::vector< double > &v, std::vector< double > const &min, std::vector< double > const &max)
- void [SDH::ToRange](#) (cSimpleVector &v, std::vector< double > const &min, std::vector< double > const &max)
- double [SDH::Approx](#) (double a, double b, double eps)
- bool [SDH::Approx](#) (int n, double *a, double *b, double *eps)
- double [SDH::DegToRad](#) (double d)
- double [SDH::RadToDeg](#) (double r)
- void [SDH::SleepSec](#) (double t)
- int [SDH::CompareReleases](#) (char const *rev1, char const *rev2)

compare release strings

11.82.1 Detailed Description

Implementation of auxilliary utility functions for SDHLibrary-CPP.

11.82.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.82.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.82.4 Define Documentation

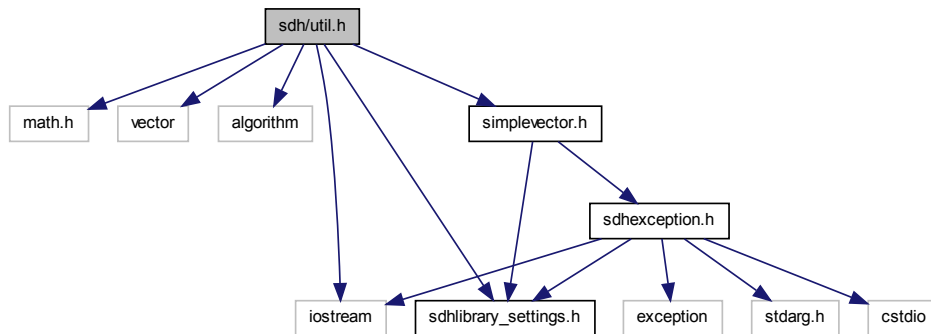
11.82.4.1 #define _USE_MATH_DEFINES

11.83 sdh/util.h File Reference

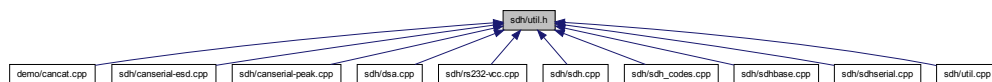
Interface of auxilliary utility functions for SDHLibrary-CPP.

```
#include <math.h>
#include <vector>
#include <algorithm>
#include <iostream>
#include "sdhlibrary_settings.h"
#include "simplevector.h"
```

Include dependency graph for util.h:



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



Classes

- class [SDH::cSetValueTemporarily< T >](#)
*helper class to set value on construction and reset to previous value on destruction.
(RAIL-idiom)*

Namespaces

- namespace [SDH](#)

Auxiliary functions

- #define [DEFINE_TO_CASECOMMAND](#)(_c) case _c: return (#_c)
- #define [DEFINE_TO_CASECOMMAND_MSG](#)(_c,...) case _c: return (#_c ": " __VA_ARGS__)
- bool [SDH::InIndex](#) (int v, int max)
- bool [SDH::InRange](#) (double v, double min, double max)

- bool [SDH::InRange](#) (int n, double const *v, double const *min, double const *max)
- double [SDH::ToRange](#) (double v, double min, double max)
- void [SDH::ToRange](#) (int n, double *v, double const *min, double const *max)
- void [SDH::ToRange](#) (std::vector< double > &v, std::vector< double > const &min, std::vector< double > const &max)
- void [SDH::ToRange](#) (cSimpleVector &v, std::vector< double > const &min, std::vector< double > const &max)
- double [SDH::Approx](#) (double a, double b, double eps)
- bool [SDH::Approx](#) (int n, double *a, double *b, double *eps)
- double [SDH::DegToRad](#) (double d)
- double [SDH::RadToDeg](#) (double r)
- void [SDH::SleepSec](#) (double t)
- template<typename Function , typename Tp >
void [SDH::apply](#) (Function f, Tp &sequence)
- template<typename Function , typename InputIterator >
Function [SDH::apply](#) (Function f, InputIterator first, InputIterator last)
- template<typename Function , typename Tp >
Tp [SDH::map](#) (Function f, Tp sequence)
- template<typename T >
std::ostream & [SDH::operator<<](#) (std::ostream &stream, std::vector< T > const &v)
- int [SDH::CompareReleases](#) (char const *rev1, char const *rev2)
compare release strings

11.83.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

11.83.2 General file information

Author

Dirk Osswald

Date

2007-02-19

11.83.3 Copyright

- Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.83.4 Define Documentation

11.83.4.1 `#define DEFINE_TO_CASECOMMAND(_c) case _c: return (#_c)`

Just a macro for the very lazy programmer to convert an enum or a DEFINE macro into a case command that returns the name of the macro as string.

Usage:

```
char const* eSomeEnumType_ToString( eSomeEnumType rc )
{
    switch (rc)
    {
        DEFINE_TO_CASECOMMAND( AN_ENUM );
        DEFINE_TO_CASECOMMAND( AN_OTHER_ENUM );
        ...
        default:
            return "unknown return code";
    }
}
```

Remarks

You must use the enum or macro directly (not a variable with that value) since CPP-stringification is used.

See also [DEFINE_TO_CASECOMMAND_MSG](#)

11.83.4.2 `#define DEFINE_TO_CASECOMMAND_MSG(_c, ...) case _c: return (#_c ": " __VA_ARGS__)`

Just a macro for the very lazy programmer to convert an enum or a DEFINE macro and a message into a case command that returns the name of the macro and the message as string.

Usage:

```
char const* eSomeEnumType_ToString( eSomeEnumType rc )
{
    switch (rc)
    {
        DEFINE_TO_CASECOMMAND_MSG( AN_ENUM, "some mighty descriptive message" );
        DEFINE_TO_CASECOMMAND_MSG( AN_OTHER_ENUM, "guess what" );
        ...
        default:
            return "unknown return code";
    }
}
```

Remarks

You must use the enum or macro directly (not a variable with that value) since CPP-stringification is used.

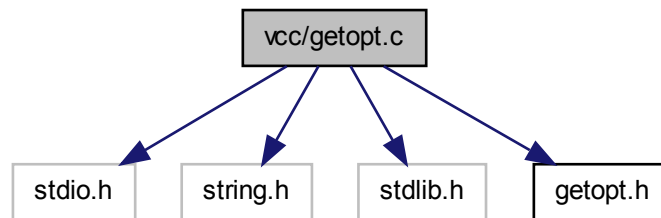
See also [DEFINE_TO_CASECOMMAND](#)

11.84 sdhlibrary_cpp.dox File Reference

11.85 vcc/getopt.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "getopt.h"
```

Include dependency graph for getopt.c:



Defines

- #define [_NO_PROTO](#)
- #define [GETOPT_INTERFACE_VERSION](#) 2
- #define [_\(msgid\)](#) (msgid)
- #define [SWAP_FLAGS](#)(ch1, ch2)
- #define [NONOPTION_P](#) (argv[optind][0] != '-' || argv[optind][1] == '\0')

Enumerations

- enum { [REQUIRE_ORDER](#), [PERMUTE](#), [RETURN_IN_ORDER](#) }

Functions

- static char * [my_index](#) (const char *str, int chr)
- static void [exchange](#) (char **argv)
- static const char * [_getopt_initialize](#) (int argc, char *const *argv, const char *optstring)

- int `_getopt_internal` (int argc, char *const *argv, const char *optstring, const struct `option` *longopts, int *longind, int long_only)
- int `getopt` (int argc, char *const *argv, const char *optstring)

Variables

- char * `optarg` = NULL
- int `optind` = 1
- int `__getopt_initialized` = 0
- static char * `nextchar`
- int `opterr` = 1
- int `optopt` = '?'
- static enum { ... } `ordering`
- static char * `posixly_correct`
- static int `first_nonopt`
- static int `last_nonopt`

11.85.1 Define Documentation

11.85.1.1 `#define _(msgid) (msgid)`

11.85.1.2 `#define _NO_PROTO`

11.85.1.3 `#define GETOPT_INTERFACE_VERSION 2`

11.85.1.4 `#define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')`

11.85.1.5 `#define SWAP_FLAGS(ch1, ch2)`

11.85.2 Enumeration Type Documentation

11.85.2.1 anonymous enum

Enumerator:

REQUIRE_ORDER

PERMUTE

RETURN_IN_ORDER

11.85.3 Function Documentation

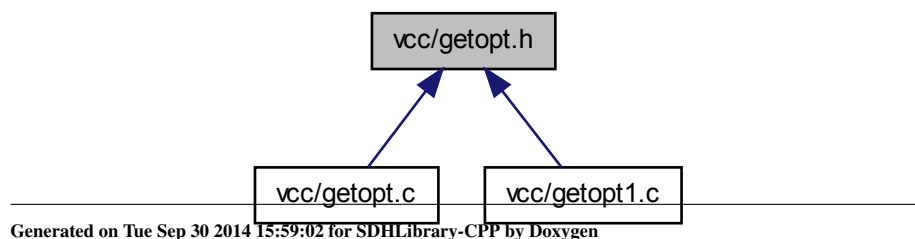
- 11.85.3.1 `static const char* _getopt_initialize (int argc, char *const * argv, const char * optstring)` [static]
- 11.85.3.2 `int _getopt_internal (int argc, char *const * argv, const char * optstring, const struct option * longopts, int * longind, int long_only)`
- 11.85.3.3 `static void exchange (char ** argv)` [static]
- 11.85.3.4 `int getopt (int argc, char *const * argv, const char * optstring)`
- 11.85.3.5 `static char* my_index (const char * str, int chr)` [static]

11.85.4 Variable Documentation

- 11.85.4.1 `int __getopt_initialized = 0`
- 11.85.4.2 `int first_nonopt` [static]
- 11.85.4.3 `int last_nonopt` [static]
- 11.85.4.4 `char* nextchar` [static]
- 11.85.4.5 `char* optarg = NULL`
- 11.85.4.6 `int opterr = 1`
- 11.85.4.7 `int optind = 1`
- 11.85.4.8 `int optopt = '?'`
- 11.85.4.9 `enum { ... } ordering` [static]
- 11.85.4.10 `char* posixly_correct` [static]

11.86 vcc/getopt.h File Reference

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



Classes

- struct [option](#)

Defines

- #define [no_argument](#) 0
- #define [required_argument](#) 1
- #define [optional_argument](#) 2

Functions

- int [getopt](#) ()
- int [getopt_long](#) (int argc, char *const *argv, const char *shortopts, const struct [option](#) *longopts, int *longind)
- int [getopt_long_only](#) (int argc, char *const *argv, const char *shortopts, const struct [option](#) *longopts, int *longind)
- int [__getopt_internal](#) (int argc, char *const *argv, const char *shortopts, const struct [option](#) *longopts, int *longind, int long_only)

Variables

- char * [optarg](#)
- int [optind](#)
- int [opterr](#)
- int [optopt](#)

11.86.1 Define Documentation

11.86.1.1 `#define no_argument 0`

11.86.1.2 `#define optional_argument 2`

11.86.1.3 `#define required_argument 1`

11.86.2 Function Documentation

11.86.2.1 `int _getopt_internal (int argc, char *const * argv, const char * shortopts, const struct option * longopts, int * longind, int long_only)`

11.86.2.2 `int getopt ()`

11.86.2.3 `int getopt_long (int argc, char *const * argv, const char * shortopts, const struct option * longopts, int * longind)`

11.86.2.4 `int getopt_long_only (int argc, char *const * argv, const char * shortopts, const struct option * longopts, int * longind)`

11.86.3 Variable Documentation

11.86.3.1 `char* optarg`

11.86.3.2 `int opterr`

11.86.3.3 `int optind`

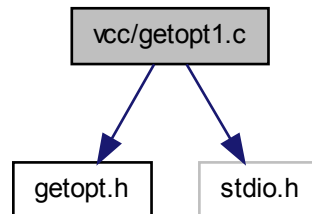
11.86.3.4 `int optopt`

11.87 vcc/getopt1.c File Reference

```
#include "getopt.h"
```

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

Include dependency graph for getopt1.c:



Defines

- #define [GETOPT_INTERFACE_VERSION](#) 2
- #define [NULL](#) 0

Functions

- int [getopt_long](#) (int argc, char *const *argv, const char *options, const struct [option](#) *long_options, int *opt_index)
- int [getopt_long_only](#) (int argc, char *const *argv, const char *options, const struct [option](#) *long_options, int *opt_index)

11.87.1 Define Documentation

11.87.1.1 #define [GETOPT_INTERFACE_VERSION](#) 2

11.87.1.2 #define [NULL](#) 0

11.87.2 Function Documentation

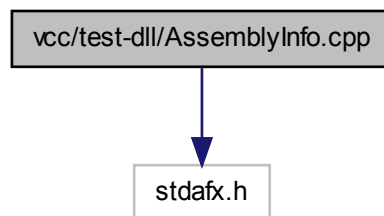
11.87.2.1 int [getopt_long](#) (int argc, char *const * argv, const char * options, const struct [option](#) * long_options, int * opt_index)

11.87.2.2 int [getopt_long_only](#) (int argc, char *const * argv, const char * options, const struct [option](#) * long_options, int * opt_index)

11.88 vcc/test-dll/AssemblyInfo.cpp File Reference

```
#include "stdafx.h"
```

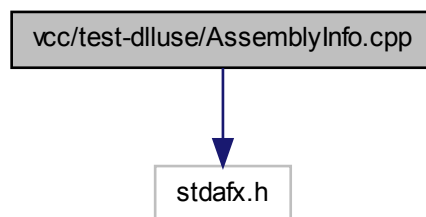

Include dependency graph for AssemblyInfo.cpp:



11.89 vcc/test-dlluse/AssemblyInfo.cpp File Reference

```
#include "stdafx.h"
```

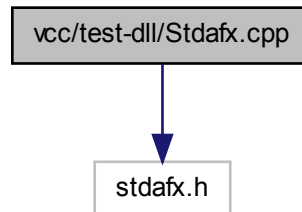
Include dependency graph for AssemblyInfo.cpp:



11.90 vcc/test-dll/Stdafx.cpp File Reference

```
#include "stdafx.h"
```

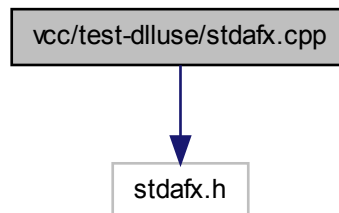
Include dependency graph for Stdafx.cpp:



11.91 vcc/test-dlluse/stdafx.cpp File Reference

```
#include "stdafx.h"
```

Include dependency graph for stdafx.cpp:



Index

- ~cCANSerial_ESD
 - SDH::cCANSerial_ESD, [74](#)
- ~cCANSerial_PEAK
 - SDH::cCANSerial_PEAK, [84](#)
- ~cDBG
 - SDH::cDBG, [102](#)
- ~cDSA
 - SDH::cDSA, [111](#)
- ~cRS232
 - SDH::cRS232, [139](#)
- ~cSDH
 - SDH::cSDH, [159](#)
- ~cSDHBase
 - SDH::cSDHBase, [233](#)
- ~cSDHOptions
 - cSDHOptions, [247](#)
- ~cSDHSerial
 - SDH::cSDHSerial, [256](#)
- ~cSerialBase
 - SDH::cSerialBase, [274](#)
- ~cSetTimeoutTemporarily
 - SDH::cSerialBase::cSetTimeoutTemporarily, [282](#)
- ~cSetValueTemporarily
 - SDH::cSetValueTemporarily, [283](#)
- - getopt.c, [479](#)
- _CRT_SECURE_NO_WARNINGS
 - rs232-vcc.cpp, [420](#)
- _GetFingerXYZ
 - SDH::cSDH, [159](#)
- _NO_PROTO
 - getopt.c, [479](#)
- _USE_MATH_DEFINES
 - demo-benchmark.cpp, [331](#)
 - sdh.cpp, [424](#)
 - util.cpp, [474](#)
- __attribute__
 - SDH, [62](#)
 - SDH::sSDHBinaryRequest, [314](#)
 - sdh_command_codes.h, [434](#)
- __author__
 - cancat.cpp, [329](#)
 - demo-benchmark.cpp, [332](#)
 - demo-contact-grasping.cpp, [334](#)
 - demo-dsa-simple.cpp, [336](#)
 - demo-dsa.cpp, [338](#)
 - demo-GetAxisActualAngle.cpp, [340](#)
 - demo-GetFingerXYZ.cpp, [342](#)
 - demo-griphand.cpp, [344](#)
 - demo-mimic.cpp, [346](#)
 - demo-radians.cpp, [348](#)
 - demo-simple-withtiming.cpp, [350](#)
 - demo-simple.cpp, [352](#)
 - demo-simple2.cpp, [354](#)
 - demo-simple3.cpp, [356](#)
 - demo-temperature.cpp, [358](#)
 - demo-velocity-acceleration.cpp, [360](#)
- __copyright__
 - cancat.cpp, [329](#)
 - demo-benchmark.cpp, [332](#)
 - demo-contact-grasping.cpp, [334](#)
 - demo-dsa-simple.cpp, [336](#)
 - demo-dsa.cpp, [338](#)
 - demo-GetAxisActualAngle.cpp, [340](#)
 - demo-GetFingerXYZ.cpp, [342](#)
 - demo-griphand.cpp, [344](#)
 - demo-mimic.cpp, [346](#)
 - demo-radians.cpp, [348](#)
 - demo-simple-withtiming.cpp, [350](#)
 - demo-simple.cpp, [352](#)
 - demo-simple2.cpp, [354](#)
 - demo-simple3.cpp, [356](#)
 - demo-temperature.cpp, [358](#)
 - demo-velocity-acceleration.cpp, [360](#)
- __getopt_initialized
 - getopt.c, [481](#)
- __help__
 - cancat.cpp, [329](#)
 - demo-benchmark.cpp, [332](#)
 - demo-contact-grasping.cpp, [334](#)
 - demo-dsa-simple.cpp, [336](#)

- demo-dsa.cpp, 338
- demo-GetAxisActualAngle.cpp, 340
- demo-GetFingerXYZ.cpp, 342
- demo-griphand.cpp, 344
- demo-mimic.cpp, 346
- demo-radians.cpp, 348
- demo-simple-withtiming.cpp, 350
- demo-simple.cpp, 352
- demo-simple2.cpp, 354
- demo-simple3.cpp, 356
- demo-temperature.cpp, 358
- demo-velocity-acceleration.cpp, 360
- __url__
 - cconcat.cpp, 329
 - demo-benchmark.cpp, 332
 - demo-contact-grasping.cpp, 334
 - demo-dsa-simple.cpp, 336
 - demo-dsa.cpp, 338
 - demo-GetAxisActualAngle.cpp, 340
 - demo-GetFingerXYZ.cpp, 342
 - demo-griphand.cpp, 344
 - demo-mimic.cpp, 346
 - demo-radians.cpp, 348
 - demo-simple-withtiming.cpp, 350
 - demo-simple.cpp, 352
 - demo-simple2.cpp, 354
 - demo-simple3.cpp, 356
 - demo-temperature.cpp, 358
 - demo-velocity-acceleration.cpp, 360
- __version__
 - cconcat.cpp, 329
 - demo-benchmark.cpp, 332
 - demo-contact-grasping.cpp, 334
 - demo-dsa-simple.cpp, 336
 - demo-dsa.cpp, 338
 - demo-GetAxisActualAngle.cpp, 340
 - demo-GetFingerXYZ.cpp, 342
 - demo-griphand.cpp, 344
 - demo-mimic.cpp, 346
 - demo-radians.cpp, 348
 - demo-simple-withtiming.cpp, 350
 - demo-simple.cpp, 352
 - demo-simple2.cpp, 354
 - demo-simple3.cpp, 356
 - demo-temperature.cpp, 358
 - demo-velocity-acceleration.cpp, 360
- _getopt_initialize
 - getopt.c, 481
- _getopt_internal
 - getopt.c, 481
- getopt.h, 483
- a
 - SDH::cSDHSerial, 256
- a_time
 - SDH::cSimpleTime, 287
- aaa
 - sRecordedData, 311
- aav
 - sRecordedData, 311
- active_interface
 - dsa.h, 396
 - SDH::cDSA::sControllerInfo, 309
- AddByte
 - SDH::cCRC, 94
- AddBytes
 - SDH::cCRC, 94
- adj_flags
 - dsa.h, 396
 - SDH::cDSA::sSensitivityInfo, 316
- alim
 - SDH::cSDHSerial, 256
- All
 - SDH::cSDHBase, 230
- all_axes
 - SDH::cSDH, 219
- all_axes_used
 - SDH::cSDHBase, 235
- all_fingers
 - SDH::cSDH, 219
- all_real_axes
 - SDH::cSDH, 219
- all_temperature_sensors
 - SDH::cSDH, 219
- apply
 - SDH, 62
- Approx
 - SDH, 62
- architecture.dox, 321
- area
 - SDH::cDSA::sContactInfo, 307
- atv
 - sRecordedData, 311
- AxisAnglesToFingerAngles
 - demo-contact-grasping.cpp, 334
- AxisCommand
 - SDH::cSDHSerial, 257
- basisdef.h
 - SDH_ASSERT_TYPESIZES, 374

- baudrate
 - SDH::cCANSerial_ESD, [75](#)
 - SDH::cCANSerial_PEAK, [87](#)
 - SDH::cRS232, [142](#)
- BaudrateToBaudrateCode
 - SDH::cCANSerial_ESD, [74](#)
 - SDH::cCANSerial_PEAK, [85](#)
 - SDH::cRS232, [139](#)
- BinaryAxisCommand
 - SDH::cSDHSerial, [257](#)
- BinarySync
 - SDH::cSDHSerial, [257](#)
- c_str
 - SDH::cMsg, [133](#)
- calib_pressure
 - SDH::cDSA, [119](#)
- calib_voltage
 - SDH::cDSA, [119](#)
- can_baudrate
 - cSDHOptions, [250](#)
 - dsa.h, [396](#)
 - SDH::cDSA::sControllerInfo, [309](#)
- CAN_ESD_RXQUEUEUSE
 - canserial-esd.h, [379](#)
- CAN_ESD_TXQUEUEUSE
 - canserial-esd.h, [379](#)
- can_id
 - dsa.h, [397](#)
 - SDH::cDSA::sControllerInfo, [309](#)
- cancat.cpp
 - __author__, [329](#)
 - __copyright__, [329](#)
 - __help__, [329](#)
 - __url__, [329](#)
 - __version__, [329](#)
 - main, [329](#)
- canserial-esd.cpp
 - DBG, [376](#)
 - ESD_strerror, [377](#)
 - SDH_CAN_SERIAL_ESD_DEBUG, [376](#)
- canserial-esd.h
 - CAN_ESD_RXQUEUEUSE, [379](#)
 - CAN_ESD_TXQUEUEUSE, [379](#)
- canserial-peak.cpp
 - DBG, [381](#)
 - M_CMSG_MSG, [381](#)
 - PEAK_strerror, [382](#)
 - SDH_CAN_SERIAL_PEAK_DEBUG, [381](#)
- USE_HANDLE, [381](#)
- USE_HANDLES, [382](#)
- cCANSerial_ESD
 - SDH::cCANSerial_ESD, [73](#)
- cCANSerial_ESDException
 - SDH::cCANSerial_ESDException, [80](#)
- cCANSerial_PEAK
 - SDH::cCANSerial_PEAK, [84](#)
- cCANSerial_PEAKException
 - SDH::cCANSerial_PEAKException, [91](#)
- cCRC
 - SDH::cCRC, [93](#)
- cCRC_DSACON32m
 - SDH::cCRC_DSACON32m, [97](#)
- cCRC_SDH
 - SDH::cCRC_SDH, [101](#)
- cDBG
 - SDH::cDBG, [102](#)
- cdbg
 - demo-benchmark.cpp, [332](#)
 - demo-contact-grasping.cpp, [334](#)
 - dsaboot.cpp, [361](#)
 - SDH::cSDHBase, [235](#)
- cDSA
 - SDH::cDSA, [110](#), [111](#)
- cDSAException
 - SDH::cDSAException, [123](#)
- cDSAUpdater, [123](#)
 - cDSAUpdater, [123](#)
 - DEFAULT_ERROR_THRESHOLD, [124](#)
 - interrupt, [124](#)
- cells_x
 - dsa.h, [397](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
- cells_y
 - dsa.h, [397](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
- CheckCRC16
 - SDH::cSDHBinaryResponse, [315](#)
 - sdhserial.cpp, [451](#)
- CheckFirmwareRelease
 - SDH::cSDH, [159](#)
- CheckIndex
 - SDH::cSDHBase, [233](#)
- CheckRange
 - SDH::cSDHBase, [233](#)
- cHexString
 - SDH::cHexString, [125](#)

- cIsGraspedBase, [125](#)
 - cIsGraspedBase, [128](#)
 - IsGrasped, [128](#)
 - ts, [128](#)
- cIsGraspedByArea, [128](#)
 - cIsGraspedByArea, [131](#)
 - IsGrasped, [131](#)
 - SetCondition, [131](#), [132](#)
- Close
 - SDH::cCANSerial_ESD, [74](#)
 - SDH::cCANSerial_PEAK, [85](#)
 - SDH::cDSA, [111](#)
 - SDH::cRS232, [139](#)
 - SDH::cSDH, [160](#)
 - SDH::cSDHSerial, [258](#)
 - SDH::cSerialBase, [275](#)
 - SDH::cTCPSerial, [297](#)
- cmd_code
 - SDH::sSDHBinaryRequest, [314](#)
 - SDH::sSDHBinaryResponse, [315](#)
 - sdhserial.cpp, [452](#)
- CMDC_A
 - sdh_command_codes.h, [434](#)
- CMDC_ALIM
 - sdh_command_codes.h, [434](#)
- CMDC_CHANGE_CHANNEL
 - sdh_command_codes.h, [434](#)
- CMDC_CHANGE_RS232
 - sdh_command_codes.h, [434](#)
- CMDC_CON
 - sdh_command_codes.h, [434](#)
- CMDC_DEBUG
 - sdh_command_codes.h, [434](#)
- CMDC_DEMO
 - sdh_command_codes.h, [434](#)
- CMDC_GET_DURATION
 - sdh_command_codes.h, [434](#)
- CMDC_GRIP
 - sdh_command_codes.h, [434](#)
- CMDC_ID
 - sdh_command_codes.h, [434](#)
- CMDC_IGRIP
 - sdh_command_codes.h, [434](#)
- CMDC_IHOLD
 - sdh_command_codes.h, [434](#)
- CMDC_ILIM
 - sdh_command_codes.h, [434](#)
- CMDC_KV
 - sdh_command_codes.h, [434](#)
- CMDC_M
 - sdh_command_codes.h, [434](#)
- CMDC_NUMAXIS
 - sdh_command_codes.h, [434](#)
- CMDC_P
 - sdh_command_codes.h, [434](#)
- CMDC_P_MAX
 - sdh_command_codes.h, [434](#)
- CMDC_P_MIN
 - sdh_command_codes.h, [434](#)
- CMDC_P_OFFSET
 - sdh_command_codes.h, [434](#)
- CMDC_PID
 - sdh_command_codes.h, [434](#)
- CMDC_POS
 - sdh_command_codes.h, [434](#)
- CMDC_POS_SAVE
 - sdh_command_codes.h, [434](#)
- CMDC_POWER
 - sdh_command_codes.h, [434](#)
- CMDC_REF
 - sdh_command_codes.h, [434](#)
- CMDC_RVEL
 - sdh_command_codes.h, [434](#)
- CMDC_SELGRIP
 - sdh_command_codes.h, [434](#)
- CMDC_SN
 - sdh_command_codes.h, [434](#)
- CMDC_SOC
 - sdh_command_codes.h, [434](#)
- CMDC_SOC_DATE
 - sdh_command_codes.h, [434](#)
- CMDC_STATE
 - sdh_command_codes.h, [434](#)
- CMDC_STOP
 - sdh_command_codes.h, [434](#)
- CMDC_TEMP
 - sdh_command_codes.h, [434](#)
- CMDC_TERMINAL
 - sdh_command_codes.h, [434](#)
- CMDC_TPAP
 - sdh_command_codes.h, [434](#)
- CMDC_TVAV
 - sdh_command_codes.h, [434](#)
- CMDC_USE_FIXED_LENGTH
 - sdh_command_codes.h, [434](#)
- CMDC_USER_ERRORS
 - sdh_command_codes.h, [434](#)
- CMDC_V
 - sdh_command_codes.h, [434](#)
- CMDC_VEL

- sdh_command_codes.h, [434](#)
- CMDC_VER
 - sdh_command_codes.h, [434](#)
- CMDC_VER_DATE
 - sdh_command_codes.h, [434](#)
- CMDC_VLIM
 - sdh_command_codes.h, [434](#)
- CMDC_VP
 - sdh_command_codes.h, [434](#)
- cMsg
 - SDH::cMsg, [133](#)
- cog_x
 - SDH::cDSA::sContactInfo, [307](#)
- cog_y
 - SDH::cDSA::sContactInfo, [308](#)
- com
 - SDH::cDSA, [119](#)
 - SDH::cSDH, [219](#)
 - SDH::cSDHSerial, [270](#)
- comm_interface
 - SDH::cSDH, [219](#)
- CompareReleases
 - SDH, [63](#)
- Compile time settings, [23](#)
- con
 - SDH::cSDHSerial, [258](#)
- connectors.dox, [324](#)
- contact_area_cell_threshold
 - SDH::cDSA, [119](#)
- contact_force_cell_threshold
 - SDH::cDSA, [119](#)
- controller_info
 - SDH::cDSA, [119](#)
- controller_type_name
 - SDH::cSDHBase, [235](#)
- controllerinfo
 - cSDHOptions, [250](#)
- CRC16
 - SDH::sSDHBinaryRequest, [313](#)
 - SDH::sSDHBinaryResponse, [315](#)
 - sdhserial.cpp, [451](#)
- crc_table
 - SDH::cCRC, [94](#)
- crc_table_dsacon32m
 - SDH::cCRC_DSACON32m, [97](#)
- cRS232
 - SDH::cRS232, [138](#)
- cRS232Exception
 - SDH::cRS232Exception, [145](#)
- cSDH
 - SDH::cSDH, [157](#)
- cSDHBase
 - SDH::cSDHBase, [233](#)
- cSDHErrorCommunication
 - SDH::cSDHErrorCommunication, [240](#)
- cSDHErrorInvalidParameter
 - SDH::cSDHErrorInvalidParameter, [242](#)
- cSDHLibraryException
 - SDH::cSDHLibraryException, [244](#)
- cSDHOptions, [245](#)
 - ~cSDHOptions, [247](#)
 - can_baudrate, [250](#)
 - controllerinfo, [250](#)
 - cSDHOptions, [247](#)
 - debug_level, [250](#)
 - debuglog, [250](#)
 - do_RLE, [250](#)
 - dsa_rs_device, [250](#)
 - dsa_tcp_port, [250](#)
 - dsa_use_tcp, [250](#)
 - dsaport, [250](#)
 - framerate, [250](#)
 - fullframe, [250](#)
 - id_read, [250](#)
 - id_write, [250](#)
 - matrixinfo, [250](#)
 - MAX_DEV_LENGTH, [250](#)
 - net, [250](#)
 - OpenCommunication, [247](#)
 - Parse, [247](#)
 - period, [250](#)
 - persistent, [250](#)
 - reset_to_default, [250](#)
 - rs232_baudrate, [250](#)
 - sdh_canpeak_device, [250](#)
 - sdh_rs_device, [250](#)
 - sdhport, [250](#)
 - sensitivity, [250](#)
 - sensorinfo, [250](#)
 - showdsasettings, [250](#)
 - tcp_adr, [250](#)
 - tcp_port, [250](#)
 - threshold, [250](#)
 - timeout, [250](#)
 - usage, [250](#)
 - use_can_esd, [250](#)
 - use_can_peak, [250](#)
 - use_fahrenheit, [250](#)
 - use_radians, [250](#)
 - use_tcp, [250](#)

- cSDHSerial
 - SDH::cSDHSerial, [256](#)
- cSerialBase
 - SDH::cSerialBase, [274](#)
- cSerialBaseException
 - SDH::cSerialBaseException, [280](#)
- cSetTimeoutTemporarily
 - SDH::cSerialBase::cSetTimeoutTemporarily, [282](#)
- cSetValueTemporarily
 - SDH::cSetTimeoutTemporarily, [283](#)
- cSimpleStringList
 - SDH::cSimpleStringList, [285](#)
- cSimpleTime
 - SDH::cSimpleTime, [287](#)
- cSimpleVector
 - SDH::cSimpleVector, [289](#), [290](#)
- cSimpleVectorException
 - SDH::cSimpleVectorException, [293](#)
- cTCPSerial
 - SDH::cTCPSerial, [297](#)
- cTCPSerialException
 - SDH::cTCPSerialException, [302](#)
- cUnitConverter
 - SDH::cUnitConverter, [303](#)
- cur_sens
 - dsa.h, [397](#)
 - SDH::cDSA::sSensitivityInfo, [316](#)
- current_crc
 - SDH::cCRC, [94](#)
- current_line
 - SDH::cSimpleStringList, [285](#)
- CurrentLine
 - SDH::cSimpleStringList, [285](#)
- d
 - SDH::cSDH, [219](#)
- DBG
 - canserial-esd.cpp, [376](#)
 - canserial-peak.cpp, [381](#)
 - rs232-cygwin.cpp, [416](#)
 - rs232-vcc.cpp, [420](#)
 - tcpserial.cpp, [466](#)
- dbg
 - SDH::cDSA, [119](#)
 - SDH::cSerialBase, [277](#)
- dbg.h
 - VAL, [389](#)
 - VAR, [389](#)
- debug
 - SDH::cSDHSerial, [258](#)
- debug_color
 - SDH::cDBG, [103](#)
- debug_flag
 - SDH::cDBG, [103](#)
- debug_level
 - cSDHOptions, [250](#)
- debuglog
 - SDH::cSDHBase, [235](#)
 - cSDHOptions, [250](#)
- decimal_places
 - SDH::cUnitConverter, [305](#)
- DEFAULT_ERROR_THRESHOLD
 - cDSAUpdater, [124](#)
- DEFINE_TO_CASECOMMAND
 - util.h, [477](#)
- DEFINE_TO_CASECOMMAND_MSG
 - util.h, [477](#)
- DegToRad
 - SDH, [63](#)
- demo
 - SDH::cSDHSerial, [258](#)
- demo-benchmark.cpp
 - _USE_MATH_DEFINES, [331](#)
 - __author__, [332](#)
 - __copyright__, [332](#)
 - __help__, [332](#)
 - __url__, [332](#)
 - __version__, [332](#)
 - cdbg, [332](#)
 - DEMO_BENCHMARK_USE_COMBINED_-SET_GET, [331](#)
 - GotoPose, [331](#)
 - main, [331](#)
 - usage, [332](#)
- demo-contact-grasping.cpp
 - __author__, [334](#)
 - __copyright__, [334](#)
 - __help__, [334](#)
 - __url__, [334](#)
 - __version__, [334](#)
 - AxisAnglesToFingerAngles, [334](#)
 - cdbg, [334](#)
 - GotoStartPose, [334](#)
 - main, [334](#)
 - usage, [334](#)
- demo-dsa-simple.cpp
 - __author__, [336](#)
 - __copyright__, [336](#)
 - __help__, [336](#)

- [__url__](#), 336
 - [__version__](#), 336
- [main](#), 335
- [usage](#), 336
- [demo-dsa.cpp](#)
 - [__author__](#), 338
 - [__copyright__](#), 338
 - [__help__](#), 338
 - [__url__](#), 338
 - [__version__](#), 338
- [main](#), 338
- [usage](#), 338
- [demo-GetAxisActualAngle.cpp](#)
 - [__author__](#), 340
 - [__copyright__](#), 340
 - [__help__](#), 340
 - [__url__](#), 340
 - [__version__](#), 340
- [main](#), 340
- [usage](#), 340
- [demo-GetFingerXYZ.cpp](#)
 - [__author__](#), 342
 - [__copyright__](#), 342
 - [__help__](#), 342
 - [__url__](#), 342
 - [__version__](#), 342
- [main](#), 342
- [usage](#), 342
- [demo-griphand.cpp](#)
 - [__author__](#), 344
 - [__copyright__](#), 344
 - [__help__](#), 344
 - [__url__](#), 344
 - [__version__](#), 344
- [main](#), 344
- [usage](#), 344
- [demo-mimic.cpp](#)
 - [__author__](#), 346
 - [__copyright__](#), 346
 - [__help__](#), 346
 - [__url__](#), 346
 - [__version__](#), 346
- [GetHand](#), 346
- [main](#), 346
- [demo-radians.cpp](#)
 - [__author__](#), 348
 - [__copyright__](#), 348
 - [__help__](#), 348
 - [__url__](#), 348
 - [__version__](#), 348
- [main](#), 348
- [usage](#), 348
- [demo-simple-withtiming.cpp](#)
 - [__author__](#), 350
 - [__copyright__](#), 350
 - [__help__](#), 350
 - [__url__](#), 350
 - [__version__](#), 350
- [main](#), 350
- [usage](#), 350
- [demo-simple.cpp](#)
 - [__author__](#), 352
 - [__copyright__](#), 352
 - [__help__](#), 352
 - [__url__](#), 352
 - [__version__](#), 352
- [main](#), 352
- [usage](#), 352
- [demo-simple2.cpp](#)
 - [__author__](#), 354
 - [__copyright__](#), 354
 - [__help__](#), 354
 - [__url__](#), 354
 - [__version__](#), 354
- [main](#), 354
- [demo-simple3.cpp](#)
 - [__author__](#), 356
 - [__copyright__](#), 356
 - [__help__](#), 356
 - [__url__](#), 356
 - [__version__](#), 356
- [main](#), 356
- [demo-temperature.cpp](#)
 - [__author__](#), 358
 - [__copyright__](#), 358
 - [__help__](#), 358
 - [__url__](#), 358
 - [__version__](#), 358
- [main](#), 358
- [demo-velocity-acceleration.cpp](#)
 - [__author__](#), 360
 - [__copyright__](#), 360
 - [__help__](#), 360
 - [__url__](#), 360
 - [__version__](#), 360
- [main](#), 360
- [usage](#), 360
- [demo/cancat.cpp](#), 328
- [demo/demo-benchmark.cpp](#), 329
- [demo/demo-contact-grasping.cpp](#), 332

- demo/demo-dsa-simple.cpp, [334](#)
- demo/demo-dsa.cpp, [336](#)
- demo/demo-GetAxisActualAngle.cpp, [338](#)
- demo/demo-GetFingerXYZ.cpp, [340](#)
- demo/demo-griphand.cpp, [342](#)
- demo/demo-mimic.cpp, [344](#)
- demo/demo-radians.cpp, [346](#)
- demo/demo-simple-withtiming.cpp, [348](#)
- demo/demo-simple.cpp, [350](#)
- demo/demo-simple2.cpp, [352](#)
- demo/demo-simple3.cpp, [354](#)
- demo/demo-temperature.cpp, [356](#)
- demo/demo-velocity-acceleration.cpp, [358](#)
- demo/dsaboost.cpp, [360](#)
- demo/dsaboost.h, [361](#)
- demo/sdhoptions.cpp, [363](#)
- demo/sdhoptions.h, [367](#)
- DEMO_BENCHMARK_USE_COMBINED_-
SET_GET
 - demo-benchmark.cpp, [331](#)
- Demonstration programs, [24](#)
- Derived compile time settings, [24](#)
- device_format_string
 - SDH::cRS232, [142](#)
- do_RLE
 - cSDHOptions, [250](#)
 - SDH::cDSA, [119](#)
- doc/onlinehelp-demo-benchmark.exe.dox, [370](#)
- doc/onlinehelp-demo-contact-grasping.exe.dox, [370](#)
- doc/onlinehelp-demo-dsa-simple.exe.dox, [370](#)
- doc/onlinehelp-demo-dsa-simple.log.dox, [370](#)
- doc/onlinehelp-demo-dsa.exe.dox, [370](#)
- doc/onlinehelp-demo-GetAxisActualAngle.exe.dox, [370](#)
- doc/onlinehelp-demo-GetFingerXYZ.exe.dox, [370](#)
- doc/onlinehelp-demo-griphand.exe.dox, [370](#)
- doc/onlinehelp-demo-mimic.exe.dox, [370](#)
- doc/onlinehelp-demo-radians.exe.dox, [370](#)
- doc/onlinehelp-demo-ref.exe.dox, [370](#)
- doc/onlinehelp-demo-simple-withtiming.exe.dox, [370](#)
- doc/onlinehelp-demo-simple.exe.dox, [370](#)
- doc/onlinehelp-demo-simple2.exe.dox, [370](#)
- doc/onlinehelp-demo-simple3.exe.dox, [370](#)
- doc/onlinehelp-demo-temperature.exe.dox, [370](#)
- doc/onlinehelp-demo-test.exe.dox, [370](#)
- doc/onlinehelp-demo-velocity-acceleration.exe.dox, [370](#)
- Doxyfile, [370](#)
- dsa.cpp
 - eDSA_ADJUST_MATRIX_SENSITIVITY, [392](#)
 - eDSA_CONFIGURE_DATA_ACQUISITION, [392](#)
 - eDSA_FULL_FRAME, [391](#)
 - eDSA_GET_MATRIX_THRESHOLD, [392](#)
 - eDSA_GET_PROPERTIES_CONTROL_-
VECTOR_OF_MATRIX, [392](#)
 - eDSA_GET_SENSITIVITY_ADJUSTMENT_-
INFO, [392](#)
 - eDSA_LOOP, [392](#)
 - eDSA_QUERY_CONTROLLER_CONFIGURATION, [391](#)
 - eDSA_QUERY_CONTROLLER_FEATURES, [392](#)
 - eDSA_QUERY_CONTROLLER_STATE, [392](#)
 - eDSA_QUERY_MATRIX_CONFIGURATION, [392](#)
 - eDSA_QUERY_SENSOR_CONFIGURATION, [391](#)
 - eDSA_READ_DESCRIPTOR_STRING, [392](#)
 - eDSA_READ_MATRIX_MASK, [392](#)
 - eDSA_SET_DYNAMIC_MASK, [392](#)
 - eDSA_SET_MATRIX_THRESHOLD, [392](#)
 - eDSA_SET_PROPERTIES_CONTROL_-
VECTOR_FOR_MATRIX, [392](#)
 - eDSA_SET_PROPERTIES_SAMPLE_-
RATE, [392](#)
 - eDSAPacketID, [391](#)
 - PRINT_MEMBER, [391](#)
 - PRINT_MEMBER_HEX, [391](#)
- dsa.h
 - active_interface, [396](#)
 - adj_flags, [396](#)
 - can_baudrate, [396](#)
 - can_id, [397](#)
 - cells_x, [397](#)
 - cells_y, [397](#)
 - cur_sens, [397](#)
 - DSA_MAX_PREAMBLE_SEARCH, [396](#)

- error_code, [397](#)
- fact_sens, [397](#)
- feature_flags, [397](#)
- fullscale, [399](#)
- generated_by, [399](#)
- hw_revision, [399](#)
- hw_version, [399](#)
- matrix_center_x, [399](#)
- matrix_center_y, [399](#)
- matrix_center_z, [399](#)
- matrix_theta_x, [399](#)
- matrix_theta_y, [399](#)
- matrix_theta_z, [399](#)
- max_payload_size, [399](#)
- nb_matrices, [399](#)
- packet_id, [399](#)
- payload, [399](#)
- reserved, [399](#)
- senscon_type, [399](#)
- serial_no, [399](#)
- size, [399](#)
- sResponse, [396](#)
- status_flags, [399](#)
- sw_version, [399](#)
- texel_height, [399](#)
- texel_width, [399](#)
- uid, [399](#)
- DSA_DEFAULT_TCP_PORT
 - sdhoptions.h, [368](#)
- DSA_MAX_PREAMBLE_SEARCH
 - dsa.h, [396](#)
- dsa_rs_device
 - cSDHOptions, [250](#)
- dsa_tcp_port
 - cSDHOptions, [250](#)
- dsa_use_tcp
 - cSDHOptions, [250](#)
- dsaboot.cpp
 - cdbg, [361](#)
- dsaport
 - cSDHOptions, [250](#)
- E_ACCESS_DENIED
 - SDH::cDSA, [110](#)
- E_ALREADY_OPEN
 - SDH::cDSA, [110](#)
- E_ALREADY_RUNNING
 - SDH::cDSA, [109](#)
- E_CHECKSUM_ERROR
 - SDH::cDSA, [110](#)
- E_CMD_ABORTED
 - SDH::cDSA, [110](#)
- E_CMD_FAILED
 - SDH::cDSA, [110](#)
- E_CMD_FORMAT_ERROR
 - SDH::cDSA, [110](#)
- E_CMD_NOT_ENOUGH_PARAMS
 - SDH::cDSA, [110](#)
- E_CMD_PENDING
 - SDH::cDSA, [110](#)
- E_CMD_UNKNOWN
 - SDH::cDSA, [110](#)
- E_DEVICE_NOT_FOUND
 - SDH::cDSA, [110](#)
- E_DEVICE_NOT_OPENED
 - SDH::cDSA, [110](#)
- E_FEATURE_NOT_SUPPORTED
 - SDH::cDSA, [109](#)
- E_INCONSISTENT_DATA
 - SDH::cDSA, [109](#)
- E_INDEX_OUT_OF_BOUNDS
 - SDH::cDSA, [110](#)
- E_INSUFFICIENT_RESOURCES
 - SDH::cDSA, [110](#)
- E_INVALID_HANDLE
 - SDH::cDSA, [110](#)
- E_INVALID_PARAMETER
 - SDH::cDSA, [110](#)
- E_IO_ERROR
 - SDH::cDSA, [110](#)
- E_NO_SENSOR
 - SDH::cDSA, [109](#)
- E_NOT_AVAILABLE
 - SDH::cDSA, [109](#)
- E_NOT_INITIALIZED
 - SDH::cDSA, [109](#)
- E_OVERRUN
 - SDH::cDSA, [110](#)
- E_RANGE_ERROR
 - SDH::cDSA, [110](#)
- E_READ_ERROR
 - SDH::cDSA, [109](#)
- E_SUCCESS
 - SDH::cDSA, [109](#)
- E_TIMEOUT
 - SDH::cDSA, [109](#)
- E_WRITE_ERROR
 - SDH::cDSA, [109](#)
- eAS_CCW_BLOCKED
 - SDH::cSDH, [157](#)

- eAS_CW_BLOCKED
SDH::cSDH, [157](#)
- eAS_DIMENSION
SDH::cSDH, [157](#)
- eAS_DISABLED
SDH::cSDH, [157](#)
- eAS_IDLE
SDH::cSDH, [157](#)
- eAS_LIMITS_REACHED
SDH::cSDH, [157](#)
- eAS_NOT_INITIALIZED
SDH::cSDH, [157](#)
- eAS_POSITIONING
SDH::cSDH, [157](#)
- eAS_SPEED_MODE
SDH::cSDH, [157](#)
- eAxisState
SDH::cSDH, [157](#)
- eCommandCode
sdh_command_codes.h, [432](#)
- eCommandCodeEnum
sdh_command_codes.h, [432](#)
- eControllerType
SDH::cSDHBase, [230](#)
- eCT_DIMENSION
SDH::cSDHBase, [231](#)
- eCT_INVALID
SDH::cSDHBase, [230](#)
- eCT_POSE
SDH::cSDHBase, [230](#)
- eCT_VELOCITY
SDH::cSDHBase, [231](#)
- eCT_VELOCITY_ACCELERATION
SDH::cSDHBase, [231](#)
- eDSA_ADJUST_MATRIX_SENSITIVITY
dsa.cpp, [392](#)
- eDSA_CONFIGURE_DATA_ACQUISITION
dsa.cpp, [392](#)
- eDSA_FULL_FRAME
dsa.cpp, [391](#)
- eDSA_GET_MATRIX_THRESHOLD
dsa.cpp, [392](#)
- eDSA_GET_PROPERTIES_CONTROL_-
VECTOR_OF_MATRIX
dsa.cpp, [392](#)
- eDSA_GET_SENSITIVITY_ADJUSTMENT_
INFO
dsa.cpp, [392](#)
- eDSA_LOOP
dsa.cpp, [392](#)
- eDSA_QUERY_CONTROLLER_CONFIGURATION
dsa.cpp, [391](#)
- eDSA_QUERY_CONTROLLER_FEATURES
dsa.cpp, [392](#)
- eDSA_QUERY_CONTROLLER_STATE
dsa.cpp, [392](#)
- eDSA_QUERY_MATRIX_CONFIGURATION
dsa.cpp, [392](#)
- eDSA_QUERY_SENSOR_CONFIGURATION
dsa.cpp, [391](#)
- eDSA_READ_DESCRIPTOR_STRING
dsa.cpp, [392](#)
- eDSA_READ_MATRIX_MASK
dsa.cpp, [392](#)
- eDSA_SET_DYNAMIC_MASK
dsa.cpp, [392](#)
- eDSA_SET_MATRIX_THRESHOLD
dsa.cpp, [392](#)
- eDSA_SET_PROPERTIES_CONTROL_-
VECTOR_FOR_MATRIX
dsa.cpp, [392](#)
- eDSA_SET_PROPERTIES_SAMPLE_RATE
dsa.cpp, [392](#)
- eDSAPacketID
dsa.cpp, [391](#)
- eEC_ACCESS_DENIED
SDH::cSDHBase, [231](#)
- eEC_ALREADY_OPEN
SDH::cSDHBase, [231](#)
- eEC_ALREADY_RUNNING
SDH::cSDHBase, [231](#)
- eEC_AXIS_DISABLED
SDH::cSDHBase, [232](#)
- eEC_CHECKSUM_ERROR
SDH::cSDHBase, [231](#)
- eEC_CMD_ABORTED
SDH::cSDHBase, [231](#)
- eEC_CMD_FAILED
SDH::cSDHBase, [231](#)
- eEC_CMD_FORMAT_ERROR
SDH::cSDHBase, [231](#)
- eEC_CMD_UNKNOWN
SDH::cSDHBase, [231](#)
- eEC_CRC_ERROR
SDH::cSDHBase, [232](#)
- eEC_DEVICE_NOT_FOUND
SDH::cSDHBase, [231](#)
- eEC_DEVICE_NOT_OPENED

- SDH::cSDHBase, [231](#)
- eEC_DIMENSION
 - SDH::cSDHBase, [232](#)
- eEC_FEATURE_NOT_SUPPORTED
 - SDH::cSDHBase, [231](#)
- eEC_HOMING_ERROR
 - SDH::cSDHBase, [232](#)
- eEC_INCONSISTENT_DATA
 - SDH::cSDHBase, [231](#)
- eEC_INDEX_OUT_OF_BOUNDS
 - SDH::cSDHBase, [232](#)
- eEC_INSUFFICIENT_RESOURCES
 - SDH::cSDHBase, [231](#)
- eEC_INTERNAL
 - SDH::cSDHBase, [232](#)
- eEC_INVALID_HANDLE
 - SDH::cSDHBase, [231](#)
- eEC_INVALID_PARAMETER
 - SDH::cSDHBase, [231](#)
- eEC_INVALID_PASSWORD
 - SDH::cSDHBase, [232](#)
- eEC_IO_ERROR
 - SDH::cSDHBase, [231](#)
- eEC_MAX_COMMANDLINE_EXCEEDED
 - SDH::cSDHBase, [232](#)
- eEC_MAX_COMMANDS_EXCEEDED
 - SDH::cSDHBase, [232](#)
- eEC_NO_COMMAND
 - SDH::cSDHBase, [232](#)
- eEC_NO_DATAPIPE
 - SDH::cSDHBase, [231](#)
- eEC_NO_PARAMS_EXPECTED
 - SDH::cSDHBase, [231](#)
- eEC_NOT_AVAILABLE
 - SDH::cSDHBase, [231](#)
- eEC_NOT_ENOUGH_PARAMS
 - SDH::cSDHBase, [231](#)
- eEC_NOT_INITIALIZED
 - SDH::cSDHBase, [231](#)
- eEC_OVER_TEMPERATURE
 - SDH::cSDHBase, [232](#)
- eEC_RANGE_ERROR
 - SDH::cSDHBase, [231](#)
- eEC_READ_ERROR
 - SDH::cSDHBase, [231](#)
- eEC_SUCCESS
 - SDH::cSDHBase, [231](#)
- eEC_TIMEOUT
 - SDH::cSDHBase, [231](#)
- eEC_UNKNOWN_ERROR
 - SDH::cSDHBase, [232](#)
- eEC_WRITE_ERROR
 - SDH::cSDHBase, [231](#)
- eErrorCode
 - SDH::cSDHBase, [231](#)
- eGID_CENTRICAL
 - SDH::cSDHBase, [232](#)
- eGID_CYLINDRICAL
 - SDH::cSDHBase, [232](#)
- eGID_DIMENSION
 - SDH::cSDHBase, [232](#)
- eGID_INVALID
 - SDH::cSDHBase, [232](#)
- eGID_PARALLEL
 - SDH::cSDHBase, [232](#)
- eGID_SPHERICAL
 - SDH::cSDHBase, [232](#)
- eGraspId
 - SDH::cSDHBase, [232](#)
- Elapsed
 - SDH::cSimpleTime, [287](#)
- Elapsed_us
 - SDH::cSimpleTime, [287](#)
- eMAX_CHARS
 - SDH::cSimpleStringList, [284](#)
- eMAX_LINES
 - SDH::cSimpleStringList, [284](#)
- eMAX_MSG
 - SDH::cMsg, [133](#)
- eMCM_DIMENSION
 - SDH::cSDH, [157](#)
- eMCM_GRIP
 - SDH::cSDH, [157](#)
- eMCM_HOLD
 - SDH::cSDH, [157](#)
- eMCM_MOVE
 - SDH::cSDH, [157](#)
- EmergencyStop
 - SDH::cSDH, [160](#)
- eMotorCurrentMode
 - SDH::cSDH, [157](#)
- eNUMBER_OF_ELEMENTS
 - SDH, [62](#)
 - SDH::cSimpleVector, [289](#)
- EOL
 - SDH::cSDHSerial, [270](#)
- eps
 - SDH::cSDHBase, [235](#)
- eps_v
 - SDH::cSDHBase, [235](#)

- eReturnCode
 - sdh_return_codes.h, 438
- eReturnCodeEnum
 - sdh_return_codes.h, 438
- error_code
 - dsa.h, 397
 - SDH::cDSA::sControllerInfo, 309
 - SDH::cDSA::sMatrixInfo, 310
 - SDH::cDSA::sSensitivityInfo, 316
 - SDH::cDSA::sSensorInfo, 317
- ErrorCodeToString
 - SDH::cDSA, 111, 112
- ESD_strerror
 - canserial-esd.cpp, 377
- eVelocityProfile
 - SDH::cSDHBase, 232
- eVP_DIMENSION
 - SDH::cSDHBase, 232
- eVP_INVALID
 - SDH::cSDHBase, 232
- eVP_RAMP
 - SDH::cSDHBase, 232
- eVP_SIN_SQUARE
 - SDH::cSDHBase, 232
- exchange
 - getopt.c, 481
- ExtractFirmwareState
 - SDH::cSDHSerial, 258
- f_max_acceleration_v
 - SDH::cSDH, 219
- f_max_angle_v
 - SDH::cSDH, 219
- f_max_motor_current_v
 - SDH::cSDH, 219
- f_max_velocity_v
 - SDH::cSDH, 220
- f_min_acceleration_v
 - SDH::cSDH, 220
- f_min_angle_v
 - SDH::cSDH, 220
- f_min_motor_current_v
 - SDH::cSDH, 220
- f_min_velocity_v
 - SDH::cSDH, 220
- f_ones_v
 - SDH::cSDH, 220
- f_zeros_v
 - SDH::cSDH, 220
- fact_sens
 - dsa.h, 397
 - SDH::cDSA::sSensitivityInfo, 316
- factor
 - SDH::cUnitConverter, 305
- fd
 - SDH::cRS232, 142
 - SDH::cTCPSerial, 298
- feature_flags
 - dsa.h, 397
 - SDH::cDSA::sControllerInfo, 309
 - SDH::cDSA::sMatrixInfo, 310
 - SDH::cDSA::sSensorInfo, 317
- finger_axis_index
 - SDH::cSDH, 220
- finger_number_of_axes
 - SDH::cSDH, 221
- firmware_error_codes
 - SDH::cSDHBase, 235
- FIRMWARE_RELEASE_RECOMMENDED
 - release.h, 401
- firmware_state
 - SDH::cSDHBase, 236
- first_nonopt
 - getopt.c, 481
- flag
 - option, 307
- flags
 - SDH::cDSA::sTactileSensorFrame, 318
- force
 - SDH::cDSA::sContactInfo, 308
- force_factor
 - SDH::cDSA, 120
- frame
 - SDH::cDSA, 120
- framerate
 - cSDHOptions, 250
- FromString
 - SDH::cSimpleVector, 290
- fullframe
 - cSDHOptions, 250
- fullscale
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- g_sdh_debug_log
 - SDH, 67
- generated_by
 - dsa.h, 399
 - SDH::cDSA::sSensorInfo, 317
- get_duration

- SDH::cSDHSerial, 258
- GetAgeOfFrame
 - SDH::cDSA, 112
- GetAxisActualAngle
 - SDH::cSDH, 161, 162
- GetAxisActualState
 - SDH::cSDH, 162, 163
- GetAxisActualVelocity
 - SDH::cSDH, 163, 164
- GetAxisEnable
 - SDH::cSDH, 164
- GetAxisLimitAcceleration
 - SDH::cSDH, 165, 166
- GetAxisLimitVelocity
 - SDH::cSDH, 166
- GetAxisMaxAcceleration
 - SDH::cSDH, 167, 168
- GetAxisMaxAngle
 - SDH::cSDH, 168
- GetAxisMaxVelocity
 - SDH::cSDH, 169
- GetAxisMinAngle
 - SDH::cSDH, 170, 171
- GetAxisMotorCurrent
 - SDH::cSDH, 171, 172
- GetAxisReferenceVelocity
 - SDH::cSDH, 173, 174
- GetAxisTargetAcceleration
 - SDH::cSDH, 174, 175
- GetAxisTargetAngle
 - SDH::cSDH, 175
- GetAxisTargetVelocity
 - SDH::cSDH, 176, 177
- GetAxisValueVector
 - SDH::cSDH, 177
- GetContactArea
 - SDH::cDSA, 112
- GetContactInfo
 - SDH::cDSA, 112
- GetController
 - SDH::cSDH, 177
- GetControllerInfo
 - SDH::cDSA, 112
- GetCRC
 - SDH::cCRC, 94
- GetCRC_HB
 - SDH::cCRC, 94
- GetCRC_LB
 - SDH::cCRC, 94
- GetDecimalPlaces
 - SDH::cUnitConverter, 304
- GetDuration
 - SDH::cSDHSerial, 258
- GetEps
 - SDH::cSDHBase, 233
- GetEpsVector
 - SDH::cSDHBase, 233
- GetErrorMessage
 - SDH::cCANSerial_ESD, 74
 - SDH::cCANSerial_PEAK, 85
 - SDH::cSerialBase, 275
- GetErrorNumber
 - SDH::cCANSerial_ESD, 74
 - SDH::cCANSerial_PEAK, 85
 - SDH::cSerialBase, 275
 - SDH::cTCPSerial, 297
- GetFactor
 - SDH::cUnitConverter, 304
- GetFingerActualAngle
 - SDH::cSDH, 178
- GetFingerAxisIndex
 - SDH::cSDH, 179
- GetFingerEnable
 - SDH::cSDH, 179, 180
- GetFingerMaxAngle
 - SDH::cSDH, 180, 181
- GetFingerMinAngle
 - SDH::cSDH, 181
- GetFingerNumberOfAxes
 - SDH::cSDH, 182
- GetFingerTargetAngle
 - SDH::cSDH, 183
- GetFingerXYZ
 - SDH::cSDH, 183
- GetFirmwareRelease
 - SDH::cSDH, 185
- GetFirmwareReleaseRecommended
 - SDH::cSDH, 185
- GetFirmwareState
 - SDH::cSDHBase, 233
- GetFlag
 - SDH::cDBG, 102
- GetFrame
 - SDH::cDSA, 112
- GetGripMaxVelocity
 - SDH::cSDH, 185
- GetHand
 - demo-mimic.cpp, 346
- GetHandle
 - SDH::cCANSerial_ESD, 74

- SDH::cCANSerial_PEAK, 85
- GetInfo
 - SDH::cSDH, 186
- GetKind
 - SDH::cUnitConverter, 304
- GetLastErrorMessage
 - SDH::cSerialBase, 275
- GetLibraryName
 - SDH::cSDH, 186
- GetLibraryRelease
 - SDH::cSDH, 186
- GetMatrixIndex
 - SDH::cDSA, 112
- GetMatrixInfo
 - SDH::cDSA, 112
- GetMatrixSensitivity
 - SDH::cDSA, 112
- GetMatrixThreshold
 - SDH::cDSA, 113
- GetMotorCurrentModeFunction
 - SDH::cSDH, 187
- GetName
 - SDH::cUnitConverter, 304
- GetNbBytesToSend
 - SDH::sSDHBinaryRequest, 313
 - sdhserial.cpp, 451
- GetNumberOfAxes
 - SDH::cSDHBase, 234
- GetNumberOfFingers
 - SDH::cSDHBase, 234
- GetNumberOfTemperatureSensors
 - SDH::cSDHBase, 234
- GetOffset
 - SDH::cUnitConverter, 304
- getopt
 - getopt.c, 481
 - getopt.h, 483
- getopt.c
 - _, 479
 - _NO_PROTO, 479
 - __getopt_initialized, 481
 - _getopt_initialize, 481
 - _getopt_internal, 481
 - exchange, 481
 - first_nonopt, 481
 - getopt, 481
 - GETOPT_INTERFACE_VERSION, 479
 - last_nonopt, 481
 - my_index, 481
 - nextchar, 481
 - NOPTION_P, 479
 - optarg, 481
 - opterr, 481
 - optind, 481
 - optopt, 481
 - ordering, 481
 - PERMUTE, 479
 - posixly_correct, 481
 - REQUIRE_ORDER, 479
 - RETURN_IN_ORDER, 479
 - SWAP_FLAGS, 479
- getopt.h
 - _getopt_internal, 483
 - getopt, 483
 - getopt_long, 483
 - getopt_long_only, 483
 - no_argument, 483
 - optarg, 483
 - opterr, 483
 - optind, 483
 - optional_argument, 483
 - optopt, 483
 - required_argument, 483
- getopt1.c
 - GETOPT_INTERFACE_VERSION, 484
 - getopt_long, 484
 - getopt_long_only, 484
 - NULL, 484
- GETOPT_INTERFACE_VERSION
 - getopt.c, 479
 - getopt1.c, 484
- getopt_long
 - getopt.h, 483
 - getopt1.c, 484
- getopt_long_only
 - getopt.h, 483
 - getopt1.c, 484
- GetSensorInfo
 - SDH::cDSA, 113
- GetStringFromControllerType
 - SDH::cSDHBase, 234
- GetStringFromErrorCode
 - SDH::cSDHBase, 234
- GetStringFromGraspId
 - SDH::cSDHBase, 234
- GetSymbol
 - SDH::cUnitConverter, 304
- GetTemperature

- SDH::cSDH, [187](#), [188](#)
- GetTexel
 - SDH::cDSA, [113](#)
- GetTimeout
 - SDH::cSerialBase, [275](#)
- GetVelocityProfile
 - SDH::cSDH, [188](#)
- GotoPose
 - demo-benchmark.cpp, [331](#)
- GotoStartPose
 - demo-contact-grasping.cpp, [334](#)
- grasp_id_name
 - SDH::cSDHBase, [236](#)
- grip
 - SDH::cSDHSerial, [258](#)
- grip_max_velocity
 - SDH::cSDH, [221](#)
- GripHand
 - SDH::cSDH, [188](#)
- h
 - SDH::cSDH, [221](#)
- has_arg
 - option, [307](#)
- hw_revision
 - dsa.h, [399](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
 - SDH::cDSA::sSensorInfo, [317](#)
- hw_version
 - dsa.h, [399](#)
 - SDH::cDSA::sControllerInfo, [309](#)
- id
 - SDH::cSDHSerial, [259](#)
- id_read
 - cSDHOptions, [250](#)
 - SDH::cCANSerial_ESD, [75](#)
 - SDH::cCANSerial_PEAK, [87](#)
- id_write
 - cSDHOptions, [250](#)
 - SDH::cCANSerial_ESD, [76](#)
 - SDH::cCANSerial_PEAK, [87](#)
- igrip
 - SDH::cSDHSerial, [259](#)
- ihold
 - SDH::cSDHSerial, [259](#)
- ilim
 - SDH::cSDHSerial, [259](#)
- InIndex
 - SDH, [63](#)
- initial_value
 - SDH::cCRC, [94](#)
- InRange
 - SDH, [63](#)
- Int16
 - SDH, [60](#)
- Int32
 - SDH, [60](#)
- Int8
 - SDH, [60](#)
- interrupt
 - cDSAUpdater, [124](#)
- INVALID_SOCKET
 - SDH::cTCPSerial, [298](#)
- io_set_old
 - SDH::cRS232, [142](#)
- IsGrasped
 - cIsGraspedBase, [128](#)
 - cIsGraspedByArea, [131](#)
- IsOpen
 - SDH::cCANSerial_ESD, [74](#)
 - SDH::cCANSerial_PEAK, [85](#)
 - SDH::cRS232, [139](#)
 - SDH::cSDH, [190](#)
 - SDH::cSDHBase, [234](#)
 - SDH::cSDHSerial, [260](#)
 - SDH::cSerialBase, [275](#)
 - SDH::cTCPSerial, [297](#)
- IsVirtualAxis
 - SDH::cSDH, [190](#)
- kind
 - SDH::cUnitConverter, [306](#)
- kv
 - SDH::cSDHSerial, [260](#)
- l1
 - SDH::cSDH, [221](#)
- l2
 - SDH::cSDH, [221](#)
- last_nonopt
 - getopt.c, [481](#)
- Length
 - SDH::cSimpleStringList, [285](#)
- line
 - SDH::cSimpleStringList, [285](#)
- m
 - SDH::cSDHSerial, [260](#)
- m_cmsg

- SDH::cCANSerial_ESD_Internal, 77
- SDH::cCANSerial_PEAK_Internal, 88
- M_CMSG_MSG
 - canserial-peak.cpp, 381
- m_cmsg_next
 - SDH::cCANSerial_ESD_Internal, 77
 - SDH::cCANSerial_PEAK_Internal, 88
- m_device
 - SDH::cCANSerial_PEAK, 87
- m_read_another
 - SDH::cDSA, 120
- m_sequetime
 - SDH::cSDHSerial, 270
- main
 - cancat.cpp, 329
 - demo-benchmark.cpp, 331
 - demo-contact-grasping.cpp, 334
 - demo-dsa-simple.cpp, 335
 - demo-dsa.cpp, 338
 - demo-GetAxisActualAngle.cpp, 340
 - demo-GetFingerXYZ.cpp, 342
 - demo-griphand.cpp, 344
 - demo-mimic.cpp, 346
 - demo-radians.cpp, 348
 - demo-simple-withtiming.cpp, 350
 - demo-simple.cpp, 352
 - demo-simple2.cpp, 354
 - demo-simple3.cpp, 356
 - demo-temperature.cpp, 358
 - demo-velocity-acceleration.cpp, 360
- Makefile, 371
- map
 - SDH, 63
- matrix_center_x
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- matrix_center_y
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- matrix_center_z
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- matrix_info
 - SDH::cDSA, 120
- matrix_theta_x
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- matrix_theta_y
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- matrix_theta_z
 - dsa.h, 399
- matrixinfo
 - cSDHOptions, 250
- max_angle_v
 - SDH::cSDHBase, 237
- MAX_DEV_LENGTH
 - cSDHOptions, 250
- max_payload_size
 - dsa.h, 399
 - SDH::cDSA::sResponse, 312
- min_angle_v
 - SDH::cSDHBase, 237
- MoveAxis
 - SDH::cSDH, 190, 192
- MoveFinger
 - SDH::cSDH, 192
- MoveHand
 - SDH::cSDH, 194
- msg
 - SDH::cMsg, 134
 - SDH::cSDHLibraryException, 245
- my_index
 - getopt.c, 481
- mywidth
 - SDH::cDBG, 103
- name
 - option, 307
 - SDH::cUnitConverter, 306
- NAMESPACE_SDH_END
 - sdhlibrary_cpp_sdhlibrary_settings_
 - h_derived_settings_group, 24
- NAMESPACE_SDH_START
 - sdhlibrary_cpp_sdhlibrary_settings_
 - h_derived_settings_group, 24
- nb_all_axes
 - SDH::cSDH, 221
- nb_cells
 - SDH::cDSA, 120
- nb_data_bytes
 - SDH::sSDHBinaryRequest, 314
 - SDH::sSDHBinaryResponse, 315
 - sdhserial.cpp, 452
- nb_lines_to_ignore
 - SDH::cSDHSerial, 270
- nb_matrices
 - dsa.h, 399
 - SDH::cDSA::sSensorInfo, 317

- nb_valid_parameters
 - SDH::sSDHBinaryRequest, [314](#)
 - SDH::sSDHBinaryResponse, [315](#)
 - sdhserial.cpp, [452](#)
- net
 - cSDHOptions, [250](#)
 - SDH::cCANSerial_ESD, [76](#)
- nextchar
 - getopt.c, [481](#)
- NextLine
 - SDH::cSimpleStringList, [285](#)
- no_argument
 - getopt.h, [483](#)
- NONOPTION_P
 - getopt.c, [479](#)
- normal_color
 - SDH::cDBG, [103](#)
- NS_SDH
 - sdhlibrary_cpp_sdhlibrary_settings_ - h_derived_settings_group, [24](#)
- ntcan_handle
 - SDH::cCANSerial_ESD_Internal, [77](#)
- NULL
 - getopt1.c, [484](#)
- numaxis
 - SDH::cSDHSerial, [261](#)
- NUMBER_OF_AXES
 - SDH::cSDHBase, [237](#)
- NUMBER_OF_AXES_PER_FINGER
 - SDH::cSDH, [221](#)
- NUMBER_OF_FINGERS
 - SDH::cSDHBase, [237](#)
- NUMBER_OF_TEMPERATURE_SENSORS
 - SDH::cSDHBase, [237](#)
- NUMBER_OF_VIRTUAL_AXES
 - SDH::cSDH, [221](#)
- NumerifyRelease
 - SDH, [64](#)
- offset
 - SDH::cSDH, [221](#)
 - SDH::cUnitConverter, [306](#)
- ones_v
 - SDH::cSDH, [221](#)
- Online help of demonstration programs, [26](#)
- Open
 - SDH::cCANSerial_ESD, [74](#)
 - SDH::cCANSerial_PEAK, [86](#)
 - SDH::cDSA, [113](#)
 - SDH::cRS232, [139](#), [140](#)
 - SDH::cSDHSerial, [261](#)
 - SDH::cSerialBase, [276](#)
 - SDH::cTCPSerial, [297](#)
- OpenCAN_ESD
 - SDH::cSDH, [195](#)
- OpenCAN_PEAK
 - SDH::cSDH, [196](#), [197](#)
- OpenCommunication
 - cSDHOptions, [247](#)
- OpenRS232
 - SDH::cSDH, [197](#)
- OpenTCP
 - SDH::cSDH, [198](#)
- operator<<
 - SDH, [64–66](#)
 - SDH::cDSA, [119](#)
 - SDH::cHexString, [125](#)
- optarg
 - getopt.c, [481](#)
 - getopt.h, [483](#)
- opterr
 - getopt.c, [481](#)
 - getopt.h, [483](#)
- optind
 - getopt.c, [481](#)
 - getopt.h, [483](#)
- option, [306](#)
 - flag, [307](#)
 - has_arg, [307](#)
 - name, [307](#)
 - val, [307](#)
- Optional_argument
 - getopt.h, [483](#)
- optopt
 - getopt.c, [481](#)
 - getopt.h, [483](#)
- ordering
 - getopt.c, [481](#)
- output
 - SDH::cDBG, [103](#)
- p
 - SDH::cSDHSerial, [261](#)
- packet_id
 - dsa.h, [399](#)
 - SDH::cDSA::sResponse, [312](#)
- parameter
 - SDH::sSDHBinaryRequest, [314](#)
 - SDH::sSDHBinaryResponse, [315](#)

- sdhserial.cpp, 452
- parameter_bytes
 - SDH::sSDHBinaryRequest, 314
 - SDH::sSDHBinaryResponse, 315
- sdhserial.cpp, 452
- Parse
 - cSDHOptions, 247
- ParseFrame
 - SDH::cDSA, 114
- payload
 - dsa.h, 399
 - SDH::cDSA::sResponse, 312
- PCAN_HANDLE
 - SDH, 60
- PDM
 - SDH::cDBG, 102
- pDoubleUnitConverterFunction
 - SDH, 60
- peak_handle
 - SDH::cCANSerial_PEAK_Internal, 88
- PEAK_sterror
 - canserial-peak.cpp, 382
- period
 - cSDHOptions, 250
- PERMUTE
 - getopt.c, 479
- persistent
 - cSDHOptions, 250
- pGetFunction
 - SDH, 61
- pid
 - SDH::cSDHSerial, 262
- port
 - SDH::cRS232, 142
- pos
 - SDH::cSDHSerial, 262
- pos_save
 - SDH::cSDHSerial, 262
- posixly_correct
 - getopt.c, 481
- power
 - SDH::cSDHSerial, 263
- PRINT_MEMBER
 - dsa.cpp, 391
- PRINT_MEMBER_HEX
 - dsa.cpp, 391
- PROJECT_COPYRIGHT
 - release.h, 401
- PROJECT_DATE
 - release.h, 401
- PROJECT_NAME
 - release.h, 401
- PROJECT_RELEASE
 - release.h, 401
- property
 - SDH::cSDHSerial, 263
- pSetFunction
 - SDH, 61
- QueryControllerInfo
 - SDH::cDSA, 114
- QueryMatrixInfo
 - SDH::cDSA, 114
- QueryMatrixInfos
 - SDH::cDSA, 114
- QuerySensorInfo
 - SDH::cDSA, 114
- RadToDeg
 - SDH, 66
- rc
 - SDH::cCANSerial_ESD_Internal, 77
 - SDH::cCANSerial_PEAK_Internal, 88
- RC_ACCESS_DENIED
 - sdh_return_codes.h, 439
- RC_ALREADY_OPEN
 - sdh_return_codes.h, 439
- RC_ALREADY_RUNNING
 - sdh_return_codes.h, 439
- RC_AXIS_DISABLED
 - sdh_return_codes.h, 439
- RC_CHECKSUM_ERROR
 - sdh_return_codes.h, 439
- RC_CMD_ABORTED
 - sdh_return_codes.h, 439
- RC_CMD_FAILED
 - sdh_return_codes.h, 439
- RC_CMD_FORMAT_ERROR
 - sdh_return_codes.h, 439
- RC_CMD_UNKNOWN
 - sdh_return_codes.h, 439
- RC_CRC_ERROR
 - sdh_return_codes.h, 439
- RC_DEVICE_NOT_FOUND
 - sdh_return_codes.h, 439
- RC_DEVICE_NOT_OPENED
 - sdh_return_codes.h, 439
- RC_DIMENSION
 - sdh_return_codes.h, 439
- RC_FEATURE_NOT_SUPPORTED

- sdh_return_codes.h, [439](#)
- RC_HOMING_ERROR
 - sdh_return_codes.h, [440](#)
- RC_INCONSISTENT_DATA
 - sdh_return_codes.h, [440](#)
- RC_INDEX_OUT_OF_BOUNDS
 - sdh_return_codes.h, [440](#)
- RC_INSUFFICIENT_RESOURCES
 - sdh_return_codes.h, [440](#)
- RC_INTERNAL
 - sdh_return_codes.h, [440](#)
- RC_INVALID_HANDLE
 - sdh_return_codes.h, [440](#)
- RC_INVALID_PARAMETER
 - sdh_return_codes.h, [440](#)
- RC_INVALID_PASSWORD
 - sdh_return_codes.h, [440](#)
- RC_IO_ERROR
 - sdh_return_codes.h, [440](#)
- RC_MAX_COMMANDLINE_EXCEEDED
 - sdh_return_codes.h, [440](#)
- RC_MAX_COMMANDS_EXCEEDED
 - sdh_return_codes.h, [440](#)
- RC_NO_COMMAND
 - sdh_return_codes.h, [440](#)
- RC_NO_DATAPIPE
 - sdh_return_codes.h, [440](#)
- RC_NO_PARAMS_EXPECTED
 - sdh_return_codes.h, [441](#)
- RC_NOT_AVAILABLE
 - sdh_return_codes.h, [441](#)
- RC_NOT_ENOUGH_PARAMS
 - sdh_return_codes.h, [441](#)
- RC_NOT_INITIALIZED
 - sdh_return_codes.h, [441](#)
- RC_OK
 - sdh_return_codes.h, [441](#)
- RC_OVER_TEMPERATURE
 - sdh_return_codes.h, [441](#)
- RC_RANGE_ERROR
 - sdh_return_codes.h, [441](#)
- RC_READ_ERROR
 - sdh_return_codes.h, [441](#)
- RC_TIMEOUT
 - sdh_return_codes.h, [441](#)
- RC_UNKNOWN_ERROR
 - sdh_return_codes.h, [441](#)
- RC_WRITE_ERROR
 - sdh_return_codes.h, [441](#)
- Read
 - SDH::cCANSerial_ESD, [75](#)
 - SDH::cCANSerial_PEAK, [86](#)
 - SDH::cRS232, [140](#)
 - SDH::cSerialBase, [276](#)
 - SDH::cTCPSerial, [297](#)
- read_timeout_us
 - SDH::cDSA, [120](#)
- ReadControllerInfo
 - SDH::cDSA, [114](#)
- ReadFrame
 - SDH::cDSA, [114](#)
- readline
 - SDH::cRS232, [140](#)
 - SDH::cSerialBase, [276](#)
- ReadMatrixInfo
 - SDH::cDSA, [114](#)
- ReadResponse
 - SDH::cDSA, [115](#)
- ReadSensorInfo
 - SDH::cDSA, [115](#)
- ref
 - SDH::cSDHSerial, [263](#)
- release.h
 - FIRMWARE_RELEASE_RECOMMENDED, [401](#)
 - PROJECT_COPYRIGHT, [401](#)
 - PROJECT_DATE, [401](#)
 - PROJECT_NAME, [401](#)
 - PROJECT_RELEASE, [401](#)
- reply
 - SDH::cSDHSerial, [270](#)
- REQUIRE_ORDER
 - getopt.c, [479](#)
- required_argument
 - getopt.h, [483](#)
- reserved
 - dsa.h, [399](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
- Reset
 - SDH::cCRC, [94](#)
 - SDH::cSimpleStringList, [285](#)
- reset_to_default
 - cSDHOptions, [250](#)
- RETURN_IN_ORDER
 - getopt.c, [479](#)
- rs232-cygwin.cpp
 - DBG, [416](#)
 - SDH_RS232_CYGWIN_DEBUG, [416](#)
 - StrDupNew, [416](#)
- rs232-vcc.cpp

- `_CRT_SECURE_NO_WARNINGS`, 420
 - DBG, 420
 - SDH_RS232_VCC_DEBUG, 420
- rs232-vcc.h
 - SDH_RS232_VCC_ASYNC, 422
- rs232_baudrate
 - cSDHOptions, 250
- rvel
 - SDH::cSDHSerial, 264
- SDH, 55
 - `__attribute__`, 62
 - apply, 62
 - Approx, 62
 - CompareReleases, 63
 - DegToRad, 63
 - eNUMBER_OF_ELEMENTS, 62
 - g_sdh_debug_log, 67
 - InIndex, 63
 - InRange, 63
 - Int16, 60
 - Int32, 60
 - Int8, 60
 - map, 63
 - NumerifyRelease, 64
 - operator<<, 64–66
 - PCAN_HANDLE, 60
 - pDoubleUnitConverterFunction, 60
 - pGetFunction, 61
 - pSetFunction, 61
 - RadToDeg, 66
 - SDH__attribute__, 67
 - SDHCommandCodeToString, 66
 - SDHReturnCodeToString, 66
 - SleepSec, 66
 - tCRCValue, 61
 - tDeviceHandle, 61
 - ToRange, 66
 - tTimevalSec, 61
 - tTimevalUSec, 61
 - uc_identity, 67
 - UInt16, 61
 - UInt32, 61
 - UInt8, 61
- sdh.cpp
 - `_USE_MATH_DEFINES`, 424
- sdh/basisdef.h, 372
- sdh/canserial-esd.cpp, 375
- sdh/canserial-esd.h, 377
- sdh/canserial-peak.cpp, 379
- sdh/canserial-peak.h, 382
- sdh/crc.cpp, 384
- sdh/crc.h, 385
- sdh/dbg.h, 387
- sdh/dsa.cpp, 389
- sdh/dsa.h, 392
- sdh/release.h, 399
- sdh/rs232-cygwin.cpp, 414
- sdh/rs232-cygwin.h, 416
- sdh/rs232-vcc.cpp, 418
- sdh/rs232-vcc.h, 420
- sdh/sdh.cpp, 422
- sdh/sdh.h, 424
- sdh/sdh_codes.cpp, 426
- sdh/sdh_codes.h, 427
- sdh/sdh_command_codes.h, 429
- sdh/sdh_return_codes.h, 434
- sdh/sdhbase.cpp, 441
- sdh/sdhbase.h, 443
- sdh/sdhexception.cpp, 444
- sdh/sdhexception.h, 446
- sdh/sdhlibrary_settings.h, 447
- sdh/sdhserial.cpp, 449
- sdh/sdhserial.h, 452
- sdh/serialbase.cpp, 454
- sdh/serialbase.h, 455
- sdh/simplestringlist.cpp, 457
- sdh/simplestringlist.h, 459
- sdh/simpletime.h, 460
- sdh/simplevector.cpp, 462
- sdh/simplevector.h, 463
- sdh/tcpserial.cpp, 465
- sdh/tcpserial.h, 466
- sdh/unit_converter.cpp, 468
- sdh/unit_converter.h, 470
- sdh/util.cpp, 472
- sdh/util.h, 474
- SDH::cCANSerial_ESD, 69
 - `~cCANSerial_ESD`, 74
 - baudrate, 75
 - BaudrateToBaudrateCode, 74
 - cCANSerial_ESD, 73
 - Close, 74
 - GetErrorMessage, 74
 - GetErrorNumber, 74
 - GetHandle, 74
 - id_read, 75
 - id_write, 76
 - IsOpen, 74
 - net, 76

- Open, [74](#)
- Read, [75](#)
- SetTimeout, [75](#)
- status, [76](#)
- write, [75](#)
- SDH::cCANSerial_ESD_Internal, [76](#)
 - m_cmsg, [77](#)
 - m_cmsg_next, [77](#)
 - ntcan_handle, [77](#)
 - rc, [77](#)
 - timeout_ms, [77](#)
- SDH::cCANSerial_ESDException, [77](#)
 - cCANSerial_ESDException, [80](#)
- SDH::cCANSerial_PEAK, [80](#)
 - ~cCANSerial_PEAK, [84](#)
 - baudrate, [87](#)
 - BaudrateToBaudrateCode, [85](#)
 - cCANSerial_PEAK, [84](#)
 - Close, [85](#)
 - GetErrorMessage, [85](#)
 - GetErrorNumber, [85](#)
 - GetHandle, [85](#)
 - id_read, [87](#)
 - id_write, [87](#)
 - IsOpen, [85](#)
 - m_device, [87](#)
 - Open, [86](#)
 - Read, [86](#)
 - SetTimeout, [86](#)
 - write, [86](#)
- SDH::cCANSerial_PEAK_Internal, [87](#)
 - m_cmsg, [88](#)
 - m_cmsg_next, [88](#)
 - peak_handle, [88](#)
 - rc, [88](#)
- SDH::cCANSerial_PEAKException, [88](#)
 - cCANSerial_PEAKException, [91](#)
- SDH::cCRC, [91](#)
 - AddByte, [94](#)
 - AddBytes, [94](#)
 - cCRC, [93](#)
 - crc_table, [94](#)
 - current_crc, [94](#)
 - GetCRC, [94](#)
 - GetCRC_HB, [94](#)
 - GetCRC_LB, [94](#)
 - initial_value, [94](#)
 - Reset, [94](#)
- SDH::cCRC_DSACON32m, [95](#)
 - cCRC_DSACON32m, [97](#)
 - crc_table_dsacon32m, [97](#)
- SDH::cCRC_SDH, [98](#)
 - cCRC_SDH, [101](#)
- SDH::cDBG, [101](#)
 - ~cDBG, [102](#)
 - cDBG, [102](#)
 - debug_color, [103](#)
 - debug_flag, [103](#)
 - GetFlag, [102](#)
 - mywidth, [103](#)
 - normal_color, [103](#)
 - output, [103](#)
 - PDM, [102](#)
 - SetColor, [103](#)
 - SetFlag, [103](#)
 - SetOutput, [103](#)
- SDH::cDSA, [103](#)
 - ~cDSA, [111](#)
 - calib_pressure, [119](#)
 - calib_voltage, [119](#)
 - cDSA, [110](#), [111](#)
 - Close, [111](#)
 - com, [119](#)
 - contact_area_cell_threshold, [119](#)
 - contact_force_cell_threshold, [119](#)
 - controller_info, [119](#)
 - dbg, [119](#)
 - do_RLE, [119](#)
 - E_ACCESS_DENIED, [110](#)
 - E_ALREADY_OPEN, [110](#)
 - E_ALREADY_RUNNING, [109](#)
 - E_CHECKSUM_ERROR, [110](#)
 - E_CMD_ABORTED, [110](#)
 - E_CMD_FAILED, [110](#)
 - E_CMD_FORMAT_ERROR, [110](#)
 - E_CMD_NOT_ENOUGH_PARAMS, [110](#)
 - E_CMD_PENDING, [110](#)
 - E_CMD_UNKNOWN, [110](#)
 - E_DEVICE_NOT_FOUND, [110](#)
 - E_DEVICE_NOT_OPENED, [110](#)
 - E_FEATURE_NOT_SUPPORTED, [109](#)
 - E_INCONSISTENT_DATA, [109](#)
 - E_INDEX_OUT_OF_BOUNDS, [110](#)
 - E_INSUFFICIENT_RESOURCES, [110](#)
 - E_INVALID_HANDLE, [110](#)
 - E_INVALID_PARAMETER, [110](#)
 - E_IO_ERROR, [110](#)
 - E_NO_SENSOR, [109](#)
 - E_NOT_AVAILABLE, [109](#)

- [E_NOT_INITIALIZED](#), 109
- [E_OVERRUN](#), 110
- [E_RANGE_ERROR](#), 110
- [E_READ_ERROR](#), 109
- [E_SUCCESS](#), 109
- [E_TIMEOUT](#), 109
- [E_WRITE_ERROR](#), 109
- [eDSACodeError](#), 109
- [ErrorCodeToString](#), 111, 112
- [force_factor](#), 120
- [frame](#), 120
- [GetAgeOfFrame](#), 112
- [GetContactArea](#), 112
- [GetContactInfo](#), 112
- [GetControllerInfo](#), 112
- [GetFrame](#), 112
- [GetMatrixIndex](#), 112
- [GetMatrixInfo](#), 112
- [GetMatrixSensitivity](#), 112
- [GetMatrixThreshold](#), 113
- [GetSensorInfo](#), 113
- [GetTexel](#), 113
- [m_read_another](#), 120
- [matrix_info](#), 120
- [nb_cells](#), 120
- [Open](#), 113
- [operator<<](#), 119
- [ParseFrame](#), 114
- [QueryControllerInfo](#), 114
- [QueryMatrixInfo](#), 114
- [QueryMatrixInfos](#), 114
- [QuerySensorInfo](#), 114
- [read_timeout_us](#), 120
- [ReadControllerInfo](#), 114
- [ReadFrame](#), 114
- [ReadMatrixInfo](#), 114
- [ReadResponse](#), 115
- [ReadSensorInfo](#), 115
- [SDH__attribute__](#), 115, 116, 120
- [sensor_info](#), 120
- [SetFramerate](#), 116
- [SetFramerateRetries](#), 117
- [SetMatrixSensitivity](#), 117
- [SetMatrixThreshold](#), 118
- [start_dsa](#), 120
- [start_pc](#), 121
- [texel_offset](#), 121
- [tTexel](#), 109
- [UpdateFrame](#), 118
- [WriteCommand](#), 118
- [WriteCommandWithPayload](#), 119
- [SDH::cDSA::sContactInfo](#), 307
 - [area](#), 307
 - [cog_x](#), 307
 - [cog_y](#), 308
 - [force](#), 308
- [SDH::cDSA::sControllerInfo](#), 308
 - [active_interface](#), 309
 - [can_baudrate](#), 309
 - [can_id](#), 309
 - [error_code](#), 309
 - [feature_flags](#), 309
 - [hw_version](#), 309
 - [senscon_type](#), 309
 - [serial_no](#), 309
 - [status_flags](#), 309
 - [sw_version](#), 309
- [SDH::cDSA::sMatrixInfo](#), 309
 - [cells_x](#), 310
 - [cells_y](#), 310
 - [error_code](#), 310
 - [feature_flags](#), 310
 - [fullscale](#), 310
 - [hw_revision](#), 310
 - [matrix_center_x](#), 310
 - [matrix_center_y](#), 310
 - [matrix_center_z](#), 310
 - [matrix_theta_x](#), 310
 - [matrix_theta_y](#), 310
 - [matrix_theta_z](#), 310
 - [reserved](#), 310
 - [texel_height](#), 310
 - [texel_width](#), 310
 - [uid](#), 310
- [SDH::cDSA::sResponse](#), 311
 - [max_payload_size](#), 312
 - [packet_id](#), 312
 - [payload](#), 312
 - [size](#), 312
 - [sResponse](#), 312
- [SDH::cDSA::sSensitivityInfo](#), 315
 - [adj_flags](#), 316
 - [cur_sens](#), 316
 - [error_code](#), 316
 - [fact_sens](#), 316
- [SDH::cDSA::sSensorInfo](#), 317
 - [error_code](#), 317
 - [feature_flags](#), 317
 - [generated_by](#), 317
 - [hw_revision](#), 317

- [nb_matrices](#), 317
 - [serial_no](#), 317
- [SDH::cDSA::sTactileSensorFrame](#), 317
 - [flags](#), 318
 - [sTactileSensorFrame](#), 318
 - [texel](#), 318
 - [timestamp](#), 319
- [SDH::cDSAException](#), 121
 - [cDSAException](#), 123
- [SDH::cHexString](#), 124
 - [cHexString](#), 125
 - [operator<](#), 125
- [SDH::cMsg](#), 132
 - [c_str](#), 133
 - [cMsg](#), 133
 - [eMAX_MSG](#), 133
 - [msg](#), 134
- [SDH::cRS232](#), 134
 - [~cRS232](#), 139
 - [baudrate](#), 142
 - [BaudrateToBaudrateCode](#), 139
 - [Close](#), 139
 - [cRS232](#), 138
 - [device_format_string](#), 142
 - [fd](#), 142
 - [io_set_old](#), 142
 - [IsOpen](#), 139
 - [Open](#), 139, 140
 - [port](#), 142
 - [Read](#), 140
 - [readline](#), 140
 - [SetTimeout](#), 141
 - [status](#), 142
 - [UseCRC16](#), 141
 - [write](#), 141
- [SDH::cRS232Exception](#), 142
 - [cRS232Exception](#), 145
- [SDH::cSDH](#), 145
 - [~cSDH](#), 159
 - [_GetFingerXYZ](#), 159
 - [all_axes](#), 219
 - [all_fingers](#), 219
 - [all_real_axes](#), 219
 - [all_temperature_sensors](#), 219
 - [CheckFirmwareRelease](#), 159
 - [Close](#), 160
 - [com](#), 219
 - [comm_interface](#), 219
 - [cSDH](#), 157
 - [d](#), 219
 - [eAS_CCW_BLOCKED](#), 157
 - [eAS_CW_BLOCKED](#), 157
 - [eAS_DIMENSION](#), 157
 - [eAS_DISABLED](#), 157
 - [eAS_IDLE](#), 157
 - [eAS_LIMITS_REACHED](#), 157
 - [eAS_NOT_INITIALIZED](#), 157
 - [eAS_POSITIONING](#), 157
 - [eAS_SPEED_MODE](#), 157
 - [eAxisState](#), 157
 - [eMCM_DIMENSION](#), 157
 - [eMCM_GRIP](#), 157
 - [eMCM_HOLD](#), 157
 - [eMCM_MOVE](#), 157
 - [EmergencyStop](#), 160
 - [eMotorCurrentMode](#), 157
 - [f_max_acceleration_v](#), 219
 - [f_max_angle_v](#), 219
 - [f_max_motor_current_v](#), 219
 - [f_max_velocity_v](#), 220
 - [f_min_acceleration_v](#), 220
 - [f_min_angle_v](#), 220
 - [f_min_motor_current_v](#), 220
 - [f_min_velocity_v](#), 220
 - [f_ones_v](#), 220
 - [f_zeros_v](#), 220
 - [finger_axis_index](#), 220
 - [finger_number_of_axes](#), 221
 - [GetAxisActualAngle](#), 161, 162
 - [GetAxisActualState](#), 162, 163
 - [GetAxisActualVelocity](#), 163, 164
 - [GetAxisEnable](#), 164
 - [GetAxisLimitAcceleration](#), 165, 166
 - [GetAxisLimitVelocity](#), 166
 - [GetAxisMaxAcceleration](#), 167, 168
 - [GetAxisMaxAngle](#), 168
 - [GetAxisMaxVelocity](#), 169
 - [GetAxisMinAngle](#), 170, 171
 - [GetAxisMotorCurrent](#), 171, 172
 - [GetAxisReferenceVelocity](#), 173, 174
 - [GetAxisTargetAcceleration](#), 174, 175
 - [GetAxisTargetAngle](#), 175
 - [GetAxisTargetVelocity](#), 176, 177
 - [GetAxisValueVector](#), 177
 - [GetController](#), 177
 - [GetFingerActualAngle](#), 178
 - [GetFingerAxisIndex](#), 179
 - [GetFingerEnable](#), 179, 180
 - [GetFingerMaxAngle](#), 180, 181
 - [GetFingerMinAngle](#), 181

- GetFingerNumberOfAxes, 182
- GetFingerTargetAngle, 183
- GetFingerXYZ, 183
- GetFirmwareRelease, 185
- GetFirmwareReleaseRecommended, 185
- GetGripMaxVelocity, 185
- GetInfo, 186
- GetLibraryName, 186
- GetLibraryRelease, 186
- GetMotorCurrentModeFunction, 187
- GetTemperature, 187, 188
- GetVelocityProfile, 188
- grip_max_velocity, 221
- GripHand, 188
- h, 221
- IsOpen, 190
- IsVirtualAxis, 190
- l1, 221
- l2, 221
- MoveAxis, 190, 192
- MoveFinger, 192
- MoveHand, 194
- nb_all_axes, 221
- NUMBER_OF_AXES_PER_FINGER, 221
- NUMBER_OF_VIRTUAL_AXES, 221
- offset, 221
- ones_v, 221
- OpenCAN_ESD, 195
- OpenCAN_PEAK, 196, 197
- OpenRS232, 197
- OpenTCP, 198
- SetAxisEnable, 198–200
- SetAxisMotorCurrent, 200
- SetAxisTargetAcceleration, 202
- SetAxisTargetAngle, 203
- SetAxisTargetGetAxisActualAngle, 205
- SetAxisTargetGetAxisActualVelocity, 206
- SetAxisTargetVelocity, 207, 208
- SetAxisValueVector, 209
- SetController, 210
- SetDebugOutput, 211
- SetFingerEnable, 212, 213
- SetFingerTargetAngle, 213, 214
- SetVelocityProfile, 214
- Stop, 215
- ToIndexVector, 215
- uc_angle, 221
- uc_angle_degrees, 222
- uc_angle_radians, 222
- uc_angular_acceleration, 222
- uc_angular_acceleration_degrees_per_second_squared, 222
- uc_angular_acceleration_radians_per_second_squared, 222
- uc_angular_velocity, 222
- uc_angular_velocity_degrees_per_second, 222
- uc_angular_velocity_radians_per_second, 222
- uc_motor_current, 222
- uc_motor_current_ampere, 223
- uc_motor_current_milliampere, 223
- uc_position, 223
- uc_position_meter, 223
- uc_position_millimeter, 223
- uc_temperature, 223
- uc_temperature_celsius, 223
- uc_temperature_fahrenheit, 223
- uc_time, 223
- uc_time_milliseconds, 223
- uc_time_seconds, 224
- UseDegrees, 216
- UseRadians, 216
- WaitAxis, 216
- zeros_v, 224
- SDH::cSDHBase, 224
- ~cSDHBase, 233
- All, 230
- all_axes_used, 235
- cdbg, 235
- CheckIndex, 233
- CheckRange, 233
- controller_type_name, 235
- cSDHBase, 233
- debug_level, 235
- eControllerType, 230
- eCT_DIMENSION, 231
- eCT_INVALID, 230
- eCT_POSE, 230
- eCT_VELOCITY, 231
- eCT_VELOCITY_ACCELERATION, 231
- eEC_ACCESS_DENIED, 231
- eEC_ALREADY_OPEN, 231
- eEC_ALREADY_RUNNING, 231
- eEC_AXIS_DISABLED, 232
- eEC_CHECKSUM_ERROR, 231
- eEC_CMD_ABORTED, 231

- eEC_CMD_FAILED, [231](#)
- eEC_CMD_FORMAT_ERROR, [231](#)
- eEC_CMD_UNKNOWN, [231](#)
- eEC_CRC_ERROR, [232](#)
- eEC_DEVICE_NOT_FOUND, [231](#)
- eEC_DEVICE_NOT_OPENED, [231](#)
- eEC_DIMENSION, [232](#)
- eEC_FEATURE_NOT_SUPPORTED, [231](#)
- eEC_HOMING_ERROR, [232](#)
- eEC_INCONSISTENT_DATA, [231](#)
- eEC_INDEX_OUT_OF_BOUNDS, [232](#)
- eEC_INSUFFICIENT_RESOURCES, [231](#)
- eEC_INTERNAL, [232](#)
- eEC_INVALID_HANDLE, [231](#)
- eEC_INVALID_PARAMETER, [231](#)
- eEC_INVALID_PASSWORD, [232](#)
- eEC_IO_ERROR, [231](#)
- eEC_MAX_COMMANDLINE_EXCEEDED, [232](#)
- eEC_MAX_COMMANDS_EXCEEDED, [232](#)
- eEC_NO_COMMAND, [232](#)
- eEC_NO_DATAPIPE, [231](#)
- eEC_NO_PARAMS_EXPECTED, [231](#)
- eEC_NOT_AVAILABLE, [231](#)
- eEC_NOT_ENOUGH_PARAMS, [231](#)
- eEC_NOT_INITIALIZED, [231](#)
- eEC_OVER_TEMPERATURE, [232](#)
- eEC_RANGE_ERROR, [231](#)
- eEC_READ_ERROR, [231](#)
- eEC_SUCCESS, [231](#)
- eEC_TIMEOUT, [231](#)
- eEC_UNKNOWN_ERROR, [232](#)
- eEC_WRITE_ERROR, [231](#)
- eErrorCode, [231](#)
- eGID_CENTRICAL, [232](#)
- eGID_CYLINDRICAL, [232](#)
- eGID_DIMENSION, [232](#)
- eGID_INVALID, [232](#)
- eGID_PARALLEL, [232](#)
- eGID_SPHERICAL, [232](#)
- eGraspId, [232](#)
- eps, [235](#)
- eps_v, [235](#)
- eVelocityProfile, [232](#)
- eVP_DIMENSION, [232](#)
- eVP_INVALID, [232](#)
- eVP_RAMP, [232](#)
- eVP_SIN_SQUARE, [232](#)
- firmware_error_codes, [235](#)
- firmware_state, [236](#)
- GetEps, [233](#)
- GetEpsVector, [233](#)
- GetFirmwareState, [233](#)
- GetNumberOfAxes, [234](#)
- GetNumberOfFingers, [234](#)
- GetNumberOfTemperatureSensors, [234](#)
- GetStringFromControllerType, [234](#)
- GetStringFromErrorCode, [234](#)
- GetStringFromGraspId, [234](#)
- grasp_id_name, [236](#)
- IsOpen, [234](#)
- max_angle_v, [237](#)
- min_angle_v, [237](#)
- NUMBER_OF_AXES, [237](#)
- NUMBER_OF_FINGERS, [237](#)
- NUMBER_OF_TEMPERATURE_SENSORS, [237](#)
- SetDebugOutput, [234](#)
- SDH::cSDHErrorCommunication, [238](#)
- cSDHErrorCommunication, [240](#)
- SDH::cSDHErrorInvalidParameter, [240](#)
- cSDHErrorInvalidParameter, [242](#)
- SDH::cSDHLibraryException, [242](#)
- cSDHLibraryException, [244](#)
- msg, [245](#)
- what, [245](#)
- SDH::cSDHSerial, [251](#)
- ~cSDHSerial, [256](#)
- a, [256](#)
- alim, [256](#)
- AxisCommand, [257](#)
- BinaryAxisCommand, [257](#)
- BinarySync, [257](#)
- Close, [258](#)
- com, [270](#)
- con, [258](#)
- cSDHSerial, [256](#)
- debug, [258](#)
- demo, [258](#)
- EOL, [270](#)
- ExtractFirmwareState, [258](#)
- get_duration, [258](#)
- GetDuration, [258](#)
- grip, [258](#)
- id, [259](#)
- igrip, [259](#)
- ihold, [259](#)

- ilim, [259](#)
- IsOpen, [260](#)
- kv, [260](#)
- m, [260](#)
- m_sequetime, [270](#)
- nb_lines_to_ignore, [270](#)
- numaxis, [261](#)
- Open, [261](#)
- p, [261](#)
- pid, [262](#)
- pos, [262](#)
- pos_save, [262](#)
- power, [263](#)
- property, [263](#)
- ref, [263](#)
- reply, [270](#)
- rvel, [264](#)
- selgrip, [264](#)
- Send, [265](#)
- sn, [265](#)
- soc, [265](#)
- soc_date, [265](#)
- sSDHBinaryRequest, [270](#)
- sSDHBinaryResponse, [270](#)
- state, [266](#)
- stop, [266](#)
- Sync, [266](#)
- SyncUnknown, [266](#)
- temp, [266](#)
- temp_electronics, [266](#)
- terminal, [266](#)
- tpap, [267](#)
- tvav, [267](#)
- user_errors, [267](#)
- v, [268](#)
- vel, [268](#)
- ver, [268](#)
- ver_date, [269](#)
- vlim, [269](#)
- vp, [269](#)
- SDH::cSerialBase, [270](#)
- ~cSerialBase, [274](#)
- Close, [275](#)
- cSerialBase, [274](#)
- dbg, [277](#)
- GetErrorMessage, [275](#)
- GetErrorNumber, [275](#)
- GetLastErrorMessage, [275](#)
- GetTimeout, [275](#)
- IsOpen, [275](#)
- Open, [276](#)
- Read, [276](#)
- readline, [276](#)
- SetTimeout, [276](#)
- tErrorCode, [274](#)
- timeout, [277](#)
- ungetch, [277](#)
- ungetch_valid, [277](#)
- UseCRC16, [277](#)
- write, [277](#)
- SDH::cSerialBase::cSetTimeoutTemporarily, [280](#)
- ~cSetTimeoutTemporarily, [282](#)
- cSetTimeoutTemporarily, [282](#)
- SDH::cSerialBaseException, [278](#)
- cSerialBaseException, [280](#)
- SDH::cSetValueTemporarily, [282](#)
- ~cSetValueTemporarily, [283](#)
- cSetValueTemporarily, [283](#)
- SDH::cSimpleStringList, [283](#)
- cSimpleStringList, [285](#)
- current_line, [285](#)
- CurrentLine, [285](#)
- eMAX_CHARS, [284](#)
- eMAX_LINES, [284](#)
- Length, [285](#)
- line, [285](#)
- NextLine, [285](#)
- Reset, [285](#)
- SDH::cSimpleTime, [286](#)
- a_time, [287](#)
- cSimpleTime, [287](#)
- Elapsed, [287](#)
- Elapsed_us, [287](#)
- StoreNow, [287](#)
- Timeval, [287](#)
- SDH::cSimpleVector, [288](#)
- cSimpleVector, [289, 290](#)
- eNUMBER_OF_ELEMENTS, [289](#)
- FromString, [290](#)
- Valid, [290](#)
- valid, [290](#)
- value, [290](#)
- x, [290](#)
- y, [290](#)
- z, [290](#)
- SDH::cSimpleVectorException, [291](#)
- cSimpleVectorException, [293](#)
- SDH::cTCPSerial, [293](#)
- Close, [297](#)

- cTCPSerial, [297](#)
- fd, [298](#)
- GetErrorNumber, [297](#)
- INVALID_SOCKET, [298](#)
- IsOpen, [297](#)
- Open, [297](#)
- Read, [297](#)
- SetTimeout, [298](#)
- tcp_adr, [299](#)
- tcp_port, [299](#)
- TIMEOUT_RETURN_IMMEDIATELY, [299](#)
- TIMEOUT_RETURN_IMMEDIATELY_S, [299](#)
- TIMEOUT_RETURN_IMMEDIATELY_US, [299](#)
- TIMEOUT_WAIT_FOR_EVER_S, [299](#)
- TIMEOUT_WAIT_FOR_EVER_US, [299](#)
- write, [298](#)
- SDH::cTCPSerialException, [299](#)
- cTCPSerialException, [302](#)
- SDH::cUnitConverter, [302](#)
- cUnitConverter, [303](#)
- decimal_places, [305](#)
- factor, [305](#)
- GetDecimalPlaces, [304](#)
- GetFactor, [304](#)
- GetKind, [304](#)
- GetName, [304](#)
- GetOffset, [304](#)
- GetSymbol, [304](#)
- kind, [306](#)
- name, [306](#)
- offset, [306](#)
- symbol, [306](#)
- ToExternal, [304](#), [305](#)
- ToInternal, [305](#)
- SDH::sSDHBinaryRequest, [312](#)
- __attribute__, [314](#)
- cmd_code, [314](#)
- CRC16, [313](#)
- GetNbBytesToSend, [313](#)
- nb_data_bytes, [314](#)
- nb_valid_parameters, [314](#)
- parameter, [314](#)
- parameter_bytes, [314](#)
- sSDHBinaryRequest, [313](#)
- SDH::sSDHBinaryResponse, [314](#)
- CheckCRC16, [315](#)
- cmd_code, [315](#)
- CRC16, [315](#)
- nb_data_bytes, [315](#)
- nb_valid_parameters, [315](#)
- parameter, [315](#)
- parameter_bytes, [315](#)
- status_code, [315](#)
- SDH__attribute__
- SDH, [67](#)
- SDH::cDSA, [115](#), [116](#), [120](#)
- sdh_command_codes.h, [432](#)
- sdh_return_codes.h, [438](#), [439](#)
- SDH_ASSERT_TYPESIZES
- basisdef.h, [374](#)
- sdh_canpeak_device
- cSDHOptions, [250](#)
- SDH_CANSERIAL_ESD_DEBUG
- canserial-esd.cpp, [376](#)
- SDH_CANSERIAL_PEAK_DEBUG
- canserial-peak.cpp, [381](#)
- sdh_command_codes.h
- __attribute__, [434](#)
- CMDC_A, [434](#)
- CMDC ALIM, [434](#)
- CMDC_CHANGE_CHANNEL, [434](#)
- CMDC_CHANGE_RS232, [434](#)
- CMDC_CON, [434](#)
- CMDC_DEBUG, [434](#)
- CMDC_DEMO, [434](#)
- CMDC_GET_DURATION, [434](#)
- CMDC_GRIP, [434](#)
- CMDC_ID, [434](#)
- CMDC_IGRIP, [434](#)
- CMDC_IHOLD, [434](#)
- CMDC_ILIM, [434](#)
- CMDC_KV, [434](#)
- CMDC_M, [434](#)
- CMDC_NUMAXIS, [434](#)
- CMDC_P, [434](#)
- CMDC_P_MAX, [434](#)
- CMDC_P_MIN, [434](#)
- CMDC_P_OFFSET, [434](#)
- CMDC_PID, [434](#)
- CMDC_POS, [434](#)
- CMDC_POS_SAVE, [434](#)
- CMDC_POWER, [434](#)
- CMDC_REF, [434](#)
- CMDC_RVEL, [434](#)
- CMDC_SELGRIP, [434](#)
- CMDC_SN, [434](#)
- CMDC_SOC, [434](#)
- CMDC_SOC_DATE, [434](#)

- CMDC_STATE, [434](#)
- CMDC_STOP, [434](#)
- CMDC_TEMP, [434](#)
- CMDC_TERMINAL, [434](#)
- CMDC_TPAP, [434](#)
- CMDC_TVAV, [434](#)
- CMDC_USE_FIXED_LENGTH, [434](#)
- CMDC_USER_ERRORS, [434](#)
- CMDC_V, [434](#)
- CMDC_VEL, [434](#)
- CMDC_VER, [434](#)
- CMDC_VER_DATE, [434](#)
- CMDC_VLIM, [434](#)
- CMDC_VP, [434](#)
- eCommandCode, [432](#)
- eCommandCodeEnum, [432](#)
- SDH__attribute__, [432](#)
- SDH_USE_VCC, [432](#)
- USE_CMD_TEST, [432](#)
- SDH_DEFAULT_TCP_ADR
 - sdhoptions.h, [368](#)
- SDH_DEFAULT_TCP_PORT
 - sdhoptions.h, [368](#)
- sdh_return_codes.h
 - eReturnCode, [438](#)
 - eReturnCodeEnum, [438](#)
 - RC_ACCESS_DENIED, [439](#)
 - RC_ALREADY_OPEN, [439](#)
 - RC_ALREADY_RUNNING, [439](#)
 - RC_AXIS_DISABLED, [439](#)
 - RC_CHECKSUM_ERROR, [439](#)
 - RC_CMD_ABORTED, [439](#)
 - RC_CMD_FAILED, [439](#)
 - RC_CMD_FORMAT_ERROR, [439](#)
 - RC_CMD_UNKNOWN, [439](#)
 - RC_CRC_ERROR, [439](#)
 - RC_DEVICE_NOT_FOUND, [439](#)
 - RC_DEVICE_NOT_OPENED, [439](#)
 - RC_DIMENSION, [439](#)
 - RC_FEATURE_NOT_SUPPORTED, [439](#)
 - RC_HOMING_ERROR, [440](#)
 - RC_INCONSISTENT_DATA, [440](#)
 - RC_INDEX_OUT_OF_BOUNDS, [440](#)
 - RC_INSUFFICIENT_RESOURCES, [440](#)
 - RC_INTERNAL, [440](#)
 - RC_INVALID_HANDLE, [440](#)
 - RC_INVALID_PARAMETER, [440](#)
 - RC_INVALID_PASSWORD, [440](#)
 - RC_IO_ERROR, [440](#)
 - RC_MAX_COMMANDLINE_EXCEEDED, [440](#)
 - RC_MAX_COMMANDS_EXCEEDED, [440](#)
 - RC_NO_COMMAND, [440](#)
 - RC_NO_DATAPIPE, [440](#)
 - RC_NO_PARAMS_EXPECTED, [441](#)
 - RC_NOT_AVAILABLE, [441](#)
 - RC_NOT_ENOUGH_PARAMS, [441](#)
 - RC_NOT_INITIALIZED, [441](#)
 - RC_OK, [441](#)
 - RC_OVER_TEMPERATURE, [441](#)
 - RC_RANGE_ERROR, [441](#)
 - RC_READ_ERROR, [441](#)
 - RC_TIMEOUT, [441](#)
 - RC_UNKNOWN_ERROR, [441](#)
 - RC_WRITE_ERROR, [441](#)
 - SDH__attribute__, [438](#), [439](#)
 - SDH_USE_VCC, [438](#)
 - SDH_RS232_CYGWIN_DEBUG
 - rs232-cygwin.cpp, [416](#)
 - SDH_RS232_VCC_ASYNC
 - rs232-vcc.h, [422](#)
 - SDH_RS232_VCC_DEBUG
 - rs232-vcc.cpp, [420](#)
 - sdh_rs_device
 - cSDHOptions, [250](#)
 - SDH_TCP_DEBUG
 - tcpserial.cpp, [466](#)
 - SDH_USE_BINARY_COMMUNICATION
 - sdhlibrary_cpp_sdhlibrary_settings_h_settings_group, [23](#)
 - SDH_USE_NAMESPACE
 - sdhlibrary_cpp_sdhlibrary_settings_h_settings_group, [23](#)
 - SDH_USE_VCC
 - sdh_command_codes.h, [432](#)
 - sdh_return_codes.h, [438](#)
 - SDHCommandCodeToString
 - SDH, [66](#)
 - sdhlibrary_cpp.dox, [478](#)
 - sdhlibrary_cpp_sdhlibrary_settings_h_derived_settings_group
 - NAMESPACE_SDH_END, [24](#)
 - NAMESPACE_SDH_START, [24](#)
 - NS_SDH, [24](#)
 - USING_NAMESPACE_SDH, [24](#)
 - sdhlibrary_cpp_sdhlibrary_settings_h_settings_group

- SDH_USE_BINARY_COMMUNICATION, [sdhoptions.cpp, 366](#)
- SDH_USE_NAMESPACE, [23](#)
- sdhoptions.cpp
 - sdhoptions_long_options, [365](#)
 - sdhoptions_short_options, [365](#)
 - sdhusage_dsaadjust, [365](#)
 - sdhusage_dsacom, [365](#)
 - sdhusage_dsaother, [366](#)
 - sdhusage_general, [366](#)
 - sdhusage_sdhcom_cancancommon, [366](#)
 - sdhusage_sdhcom_common, [366](#)
 - sdhusage_sdhcom_esdcan, [366](#)
 - sdhusage_sdhcom_peakcan, [366](#)
 - sdhusage_sdhcom_serial, [366](#)
 - sdhusage_sdhcom_tcp, [366](#)
 - sdhusage_sdhother, [366](#)
- STRINGIFY, [365](#)
- XSTRINGIFY, [365](#)
- sdhoptions.h
 - DSA_DEFAULT_TCP_PORT, [368](#)
 - SDH_DEFAULT_TCP_ADR, [368](#)
 - SDH_DEFAULT_TCP_PORT, [368](#)
 - SDHUSAGE_DEFAULT, [368](#)
- sdhoptions_long_options
 - sdhoptions.cpp, [365](#)
- sdhoptions_short_options
 - sdhoptions.cpp, [365](#)
- sdhport
 - cSDHOptions, [250](#)
- SDHReturnCodeToString
 - SDH, [66](#)
- sdhserial.cpp
 - CheckCRC16, [451](#)
 - cmd_code, [452](#)
 - CRC16, [451](#)
 - GetNbBytesToSend, [451](#)
 - nb_data_bytes, [452](#)
 - nb_valid_parameters, [452](#)
 - parameter, [452](#)
 - parameter_bytes, [452](#)
 - sSDHBinaryRequest, [451](#)
 - status_code, [452](#)
- SDHUSAGE_DEFAULT
 - sdhoptions.h, [368](#)
- sdhusage_dsaadjust
 - sdhoptions.cpp, [365](#)
- sdhusage_dsacom
 - sdhoptions.cpp, [365](#)
- sdhusage_dsaother
 - sdhoptions.cpp, [366](#)
- sdhusage_general
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_cancancommon
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_common
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_esdcan
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_peakcan
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_serial
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhcom_tcp
 - sdhoptions.cpp, [366](#)
- sdhusage_sdhother
 - sdhoptions.cpp, [366](#)
- selgrip
 - SDH::cSDHSerial, [264](#)
- Send
 - SDH::cSDHSerial, [265](#)
- senscon_type
 - dsa.h, [399](#)
 - SDH::cDSA::sControllerInfo, [309](#)
- sensitivity
 - cSDHOptions, [250](#)
- sensor_info
 - SDH::cDSA, [120](#)
- sensorinfo
 - cSDHOptions, [250](#)
- serial_no
 - dsa.h, [399](#)
 - SDH::cDSA::sControllerInfo, [309](#)
 - SDH::cDSA::sSensorInfo, [317](#)
- SetAxisEnable
 - SDH::cSDH, [198–200](#)
- SetAxisMotorCurrent
 - SDH::cSDH, [200](#)
- SetAxisTargetAcceleration
 - SDH::cSDH, [202](#)
- SetAxisTargetAngle
 - SDH::cSDH, [203](#)
- SetAxisTargetGetAxisActualAngle
 - SDH::cSDH, [205](#)
- SetAxisTargetGetAxisActualVelocity
 - SDH::cSDH, [206](#)
- SetAxisTargetVelocity
 - SDH::cSDH, [207, 208](#)
- SetAxisValueVector
 - SDH::cSDH, [209](#)

- SetColor
 - SDH::cDBG, 103
- SetCondition
 - cIsGraspedByArea, 131, 132
- SetController
 - SDH::cSDH, 210
- SetDebugOutput
 - SDH::cSDH, 211
 - SDH::cSDHBase, 234
- SetFingerEnable
 - SDH::cSDH, 212, 213
- SetFingerTargetAngle
 - SDH::cSDH, 213, 214
- SetFlag
 - SDH::cDBG, 103
- SetFramerate
 - SDH::cDSA, 116
- SetFramerateRetries
 - SDH::cDSA, 117
- SetMatrixSensitivity
 - SDH::cDSA, 117
- SetMatrixThreshold
 - SDH::cDSA, 118
- SetOutput
 - SDH::cDBG, 103
- SetTimeout
 - SDH::cCANSerial_ESD, 75
 - SDH::cCANSerial_PEAK, 86
 - SDH::cRS232, 141
 - SDH::cSerialBase, 276
 - SDH::cTCPSerial, 298
- SetVelocityProfile
 - SDH::cSDH, 214
- showsasettings
 - cSDHOptions, 250
- size
 - dsa.h, 399
 - SDH::cDSA::sResponse, 312
- SleepSec
 - SDH, 66
- sn
 - SDH::cSDHSerial, 265
- soc
 - SDH::cSDHSerial, 265
- soc_date
 - SDH::cSDHSerial, 265
- sRecordedData, 311
 - aaa, 311
 - aav, 311
 - atv, 311
 - sRecordedData, 311
 - t, 311
- sResponse
 - dsa.h, 396
 - SDH::cDSA::sResponse, 312
- sSDHBinaryRequest
 - SDH::cSDHSerial, 270
 - SDH::sSDHBinaryRequest, 313
 - sdhserial.cpp, 451
- sSDHBinaryResponse
 - SDH::cSDHSerial, 270
- sTactileSensorFrame
 - SDH::cDSA::sTactileSensorFrame, 318
- start_dsa
 - SDH::cDSA, 120
- start_pc
 - SDH::cDSA, 121
- state
 - SDH::cSDHSerial, 266
- status
 - SDH::cCANSerial_ESD, 76
 - SDH::cRS232, 142
- status_code
 - SDH::sSDHBinaryResponse, 315
 - sdhserial.cpp, 452
- status_flags
 - dsa.h, 399
 - SDH::cDSA::sControllerInfo, 309
- Stop
 - SDH::cSDH, 215
- stop
 - SDH::cSDHSerial, 266
- StoreNow
 - SDH::cSimpleTime, 287
- StrDupNew
 - rs232-cygwin.cpp, 416
- STRINGIFY
 - sdhoptions.cpp, 365
- sw_version
 - dsa.h, 399
 - SDH::cDSA::sControllerInfo, 309
- SWAP_FLAGS
 - getopt.c, 479
- symbol
 - SDH::cUnitConverter, 306
- Sync
 - SDH::cSDHSerial, 266
- SyncUnknown
 - SDH::cSDHSerial, 266

- t
 - sRecordedData, [311](#)
- tcp_adr
 - cSDHOptions, [250](#)
 - SDH::cTCPSerial, [299](#)
- tcp_port
 - cSDHOptions, [250](#)
 - SDH::cTCPSerial, [299](#)
- tcpserial.cpp
 - DBG, [466](#)
 - SDH_TCP_DEBUG, [466](#)
- tCRCValue
 - SDH, [61](#)
- tDeviceHandle
 - SDH, [61](#)
- temp
 - SDH::cSDHSerial, [266](#)
- temp_electronics
 - SDH::cSDHSerial, [266](#)
- terminal
 - SDH::cSDHSerial, [266](#)
- tErrorCode
 - SDH::cSerialBase, [274](#)
- texel
 - SDH::cDSA::sTactileSensorFrame, [318](#)
- texel_height
 - dsa.h, [399](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
- texel_offset
 - SDH::cDSA, [121](#)
- texel_width
 - dsa.h, [399](#)
 - SDH::cDSA::sMatrixInfo, [310](#)
- threshold
 - cSDHOptions, [250](#)
- timeout
 - cSDHOptions, [250](#)
 - SDH::cSerialBase, [277](#)
- timeout_ms
 - SDH::cCANSerial_ESD_Internal, [77](#)
- TIMEOUT_RETURN_IMMEDIATELY_S
 - SDH::cTCPSerial, [299](#)
- TIMEOUT_RETURN_IMMEDIATELY_US
 - SDH::cTCPSerial, [299](#)
- TIMEOUT_WAIT_FOR_EVER_S
 - SDH::cTCPSerial, [299](#)
- TIMEOUT_WAIT_FOR_EVER_US
 - SDH::cTCPSerial, [299](#)
- timestamp
 - SDH::cDSA::sTactileSensorFrame, [319](#)
- Timeval
 - SDH::cSimpleTime, [287](#)
- ToExternal
 - SDH::cUnitConverter, [304](#), [305](#)
- ToIndexVector
 - SDH::cSDH, [215](#)
- ToInternal
 - SDH::cUnitConverter, [305](#)
- ToRange
 - SDH, [66](#)
- tpap
 - SDH::cSDHSerial, [267](#)
- ts
 - cIsGraspedBase, [128](#)
- tTexel
 - SDH::cDSA, [109](#)
- tTimevalSec
 - SDH, [61](#)
- tTimevalUSec
 - SDH, [61](#)
- tvav
 - SDH::cSDHSerial, [267](#)
- uc_angle
 - SDH::cSDH, [221](#)
- uc_angle_degrees
 - SDH::cSDH, [222](#)
- uc_angle_radians
 - SDH::cSDH, [222](#)
- uc_angular_acceleration
 - SDH::cSDH, [222](#)
- uc_angular_acceleration_degrees_per_second_squared
 - SDH::cSDH, [222](#)
- uc_angular_acceleration_radians_per_second_squared
 - SDH::cSDH, [222](#)
- uc_angular_velocity
 - SDH::cSDH, [222](#)
- uc_angular_velocity_degrees_per_second
 - SDH::cSDH, [222](#)
- uc_angular_velocity_radians_per_second
 - SDH::cSDH, [222](#)
- uc_identity
 - SDH, [67](#)
- uc_motor_current
 - SDH::cSDH, [222](#)
- uc_motor_current_ampere
 - SDH::cSDH, [223](#)
- uc_motor_current_milliampere
 - SDH::cSDH, [223](#)

- SDH::cSDH, 223
- uc_position
 - SDH::cSDH, 223
- uc_position_meter
 - SDH::cSDH, 223
- uc_position_millimeter
 - SDH::cSDH, 223
- uc_temperature
 - SDH::cSDH, 223
- uc_temperature_celsius
 - SDH::cSDH, 223
- uc_temperature_fahrenheit
 - SDH::cSDH, 223
- uc_time
 - SDH::cSDH, 223
- uc_time_milliseconds
 - SDH::cSDH, 223
- uc_time_seconds
 - SDH::cSDH, 224
- uid
 - dsa.h, 399
 - SDH::cDSA::sMatrixInfo, 310
- UInt16
 - SDH, 61
- UInt32
 - SDH, 61
- UInt8
 - SDH, 61
- ungetch
 - SDH::cSerialBase, 277
- ungetch_valid
 - SDH::cSerialBase, 277
- UpdateFrame
 - SDH::cDSA, 118
- usage
 - cSDHOptions, 250
 - demo-benchmark.cpp, 332
 - demo-contact-grasping.cpp, 334
 - demo-dsa-simple.cpp, 336
 - demo-dsa.cpp, 338
 - demo-GetAxisActualAngle.cpp, 340
 - demo-GetFingerXYZ.cpp, 342
 - demo-griphand.cpp, 344
 - demo-radians.cpp, 348
 - demo-simple-withtiming.cpp, 350
 - demo-simple.cpp, 352
 - demo-velocity-acceleration.cpp, 360
- use_can_esd
 - cSDHOptions, 250
- use_can_peak
 - cSDHOptions, 250
- USE_CMD_TEST
 - sdh_command_codes.h, 432
- use_fahrenheit
 - cSDHOptions, 250
- USE_HANDLE
 - canserial-peak.cpp, 381
- USE_HANDLES
 - canserial-peak.cpp, 382
- use_radians
 - cSDHOptions, 250
- use_tcp
 - cSDHOptions, 250
- UseCRC16
 - SDH::cRS232, 141
 - SDH::cSerialBase, 277
- UseDegrees
 - SDH::cSDH, 216
- user_errors
 - SDH::cSDHSerial, 267
- UseRadians
 - SDH::cSDH, 216
- USING_NAMESPACE_SDH
 - sdhlibrary_cpp_sdhlibrary_settings_h_derived_settings_group, 24
- util.cpp
 - _USE_MATH_DEFINES, 474
- util.h
 - DEFINE_TO_CASECOMMAND, 477
 - DEFINE_TO_CASECOMMAND_MSG, 477
- v
 - SDH::cSDHSerial, 268
- VAL
 - dbg.h, 389
- val
 - option, 307
- Valid
 - SDH::cSimpleVector, 290
- valid
 - SDH::cSimpleVector, 290
- value
 - SDH::cSimpleVector, 290
- VAR
 - dbg.h, 389
- vcc/getopt.c, 478
- vcc/getopt.h, 481
- vcc/getopt1.c, 483
- vcc/test-dll/AssemblyInfo.cpp, 484

vcc/test-dll/Stdafx.cpp, [485](#)
vcc/test-dlluse/AssemblyInfo.cpp, [485](#)
vcc/test-dlluse/stdafx.cpp, [486](#)
vel
 SDH::cSDHSerial, [268](#)
ver
 SDH::cSDHSerial, [268](#)
ver_date
 SDH::cSDHSerial, [269](#)
vlim
 SDH::cSDHSerial, [269](#)
vp
 SDH::cSDHSerial, [269](#)

WaitAxis
 SDH::cSDH, [216](#)
what
 SDH::cSDHLibraryException, [245](#)
write
 SDH::cCANSerial_ESD, [75](#)
 SDH::cCANSerial_PEAK, [86](#)
 SDH::cRS232, [141](#)
 SDH::cSerialBase, [277](#)
 SDH::cTCPSerial, [298](#)
WriteCommand
 SDH::cDSA, [118](#)
WriteCommandWithPayload
 SDH::cDSA, [119](#)

x
 SDH::cSimpleVector, [290](#)
XSTRINGIFY
 sdhoptions.cpp, [365](#)

y
 SDH::cSimpleVector, [290](#)

z
 SDH::cSimpleVector, [290](#)
zeros_v
 SDH::cSDH, [224](#)