

**SDHLibrary-CPP**

0.0.1.13

Generated by Doxygen 1.5.5

Tue Feb 2 12:42:45 2010



# Contents

<b>1</b>	<b>SDHLibrary_CPP</b>	<b>1</b>
1.1	General project information . . . . .	1
1.2	Purpose . . . . .	1
1.3	Links . . . . .	1
<b>2</b>	<b>Bug List</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>9</b>
3.1	Modules . . . . .	9
<b>4</b>	<b>Namespace Index</b>	<b>11</b>
4.1	Namespace List . . . . .	11
<b>5</b>	<b>Class Index</b>	<b>13</b>
5.1	Class Hierarchy . . . . .	13
<b>6</b>	<b>Class Index</b>	<b>15</b>
6.1	Class List . . . . .	15
<b>7</b>	<b>File Index</b>	<b>17</b>
7.1	File List . . . . .	17
<b>8</b>	<b>Module Documentation</b>	<b>23</b>
8.1	Compile time settings . . . . .	23
8.2	Derived compile time settings . . . . .	24
8.3	Demonstration programs . . . . .	25
8.4	Online help of demonstration programs . . . . .	27
<b>9</b>	<b>Namespace Documentation</b>	<b>43</b>
9.1	SDH Namespace Reference . . . . .	43
<b>10</b>	<b>Class Documentation</b>	<b>53</b>

10.1 SDH::cCANSerial_ESD Class Reference . . . . .	53
10.2 SDH::cCANSerial_ESDEception Class Reference . . . . .	62
10.3 SDH::cCANSerial_PEAK Class Reference . . . . .	65
10.4 SDH::cCANSerial_PEAKException Class Reference . . . . .	73
10.5 SDH::cCRC Class Reference . . . . .	76
10.6 SDH::cCRC_DSACON32m Class Reference . . . . .	80
10.7 SDH::cDBG Class Reference . . . . .	83
10.8 SDH::cDSA Class Reference . . . . .	87
10.9 SDH::cDSA::sContactInfo Struct Reference . . . . .	104
10.10SDH::cDSA::sControllerInfo Struct Reference . . . . .	105
10.11SDH::cDSA::sMatrixInfo Struct Reference . . . . .	106
10.12SDH::cDSA::sResponse Struct Reference . . . . .	108
10.13SDH::cDSA::sSensorInfo Struct Reference . . . . .	110
10.14SDH::cDSA::sTactileSensorFrame Struct Reference . . . . .	111
10.15SDH::cDSAException Class Reference . . . . .	113
10.16SDH::cDSAUpdater Class Reference . . . . .	116
10.17SDH::cIsGraspedBase Class Reference . . . . .	120
10.18SDH::cIsGraspedByArea Class Reference . . . . .	124
10.19SDH::cMsg Class Reference . . . . .	128
10.20SDH::cRS232 Class Reference . . . . .	131
10.21SDH::cRS232Exception Class Reference . . . . .	141
10.22SDH::cSDH Class Reference . . . . .	144
10.23SDH::cSDHBase Class Reference . . . . .	268
10.24SDH::cSDHErrorCommunication Class Reference . . . . .	290
10.25SDH::cSDHErrorInvalidParameter Class Reference . . . . .	293
10.26SDH::cSDHLibraryException Class Reference . . . . .	296
10.27cSDHOPTIONS Class Reference . . . . .	300
10.28SDH::cSDHSerial Class Reference . . . . .	306
10.29SDH::cSerialBase Class Reference . . . . .	333
10.30SDH::cSerialBaseException Class Reference . . . . .	342
10.31SDH::cSimpleStringList Class Reference . . . . .	345
10.32SDH::cSimpleTime Class Reference . . . . .	349
10.33SDH::cSimpleVector Class Reference . . . . .	352
10.34SDH::cSimpleVectorException Class Reference . . . . .	357
10.35SDH::cUnitConverter Class Reference . . . . .	360
10.36option Struct Reference . . . . .	367

<b>11 File Documentation</b>	<b>369</b>
11.1 architecture.dox File Reference . . . . .	369
11.2 cpp.desire.final/architecture.dox File Reference . . . . .	372
11.3 connectors.dox File Reference . . . . .	375
11.4 cpp.desire.final/connectors.dox File Reference . . . . .	379
11.5 cpp.desire.final/demo/cancat.cpp File Reference . . . . .	383
11.6 demo/cancat.cpp File Reference . . . . .	385
11.7 cpp.desire.final/demo/demo-contact-grasping.cpp File Reference . . . . .	387
11.8 demo/demo-contact-grasping.cpp File Reference . . . . .	390
11.9 cpp.desire.final/demo/demo-dsa.cpp File Reference . . . . .	393
11.10demo/demo-dsa.cpp File Reference . . . . .	396
11.11cpp.desire.final/demo/demo-GetAxisActualAngle.cpp File Reference . . . . .	399
11.12demo/demo-GetAxisActualAngle.cpp File Reference . . . . .	402
11.13cpp.desire.final/demo/demo-GetFingerXYZ.cpp File Reference . . . . .	405
11.14demo/demo-GetFingerXYZ.cpp File Reference . . . . .	408
11.15cpp.desire.final/demo/demo-mimic.cpp File Reference . . . . .	411
11.16demo/demo-mimic.cpp File Reference . . . . .	413
11.17cpp.desire.final/demo/demo-simple-withtiming.cpp File Reference . . . . .	415
11.18demo/demo-simple-withtiming.cpp File Reference . . . . .	417
11.19cpp.desire.final/demo/demo-simple.cpp File Reference . . . . .	419
11.20demo/demo-simple.cpp File Reference . . . . .	421
11.21cpp.desire.final/demo/demo-simple2.cpp File Reference . . . . .	423
11.22demo/demo-simple2.cpp File Reference . . . . .	425
11.23cpp.desire.final/demo/demo-simple3.cpp File Reference . . . . .	427
11.24demo/demo-simple3.cpp File Reference . . . . .	429
11.25cpp.desire.final/demo/demo-temperature.cpp File Reference . . . . .	431
11.26demo/demo-temperature.cpp File Reference . . . . .	433
11.27cpp.desire.final/demo/demo-velocity-acceleration.cpp File Reference . . . . .	435
11.28demo/demo-velocity-acceleration.cpp File Reference . . . . .	437
11.29cpp.desire.final/demo/dsabooost.cpp File Reference . . . . .	439
11.30demo/dsabooost.cpp File Reference . . . . .	441
11.31cpp.desire.final/demo/dsabooost.h File Reference . . . . .	443
11.32demo/dsabooost.h File Reference . . . . .	445
11.33cpp.desire.final/demo/sdhoptions.cpp File Reference . . . . .	447
11.34demo/sdhoptions.cpp File Reference . . . . .	453
11.35cpp.desire.final/demo/sdhoptions.h File Reference . . . . .	459

11.36demo/sdhoptions.h File Reference . . . . .	461
11.37cpp.desire.final/sdh/basisdef.h File Reference . . . . .	463
11.38sdh/basisdef.h File Reference . . . . .	465
11.39cpp.desire.final/sdh/canserial-esd.cpp File Reference . . . . .	467
11.40sdh/canserial-esd.cpp File Reference . . . . .	469
11.41cpp.desire.final/sdh/canserial-esd.h File Reference . . . . .	471
11.42sdh/canserial-esd.h File Reference . . . . .	473
11.43cpp.desire.final/sdh/canserial-peak.cpp File Reference . . . . .	475
11.44sdh/canserial-peak.cpp File Reference . . . . .	477
11.45cpp.desire.final/sdh/canserial-peak.h File Reference . . . . .	479
11.46sdh/canserial-peak.h File Reference . . . . .	481
11.47cpp.desire.final/sdh/crc.cpp File Reference . . . . .	483
11.48sdh/crc.cpp File Reference . . . . .	484
11.49cpp.desire.final/sdh/crc.h File Reference . . . . .	485
11.50sdh/crc.h File Reference . . . . .	487
11.51cpp.desire.final/sdh/dbg.h File Reference . . . . .	489
11.52sdh/dbg.h File Reference . . . . .	491
11.53cpp.desire.final/sdh/dsa.cpp File Reference . . . . .	493
11.54sdh/dsa.cpp File Reference . . . . .	495
11.55cpp.desire.final/sdh/dsa.h File Reference . . . . .	497
11.56sdh/dsa.h File Reference . . . . .	500
11.57cpp.desire.final/sdh/release.h File Reference . . . . .	503
11.58sdh/release.h File Reference . . . . .	513
11.59cpp.desire.final/sdh/rs232-cygwin.cpp File Reference . . . . .	523
11.60sdh/rs232-cygwin.cpp File Reference . . . . .	525
11.61cpp.desire.final/sdh/rs232-cygwin.h File Reference . . . . .	527
11.62sdh/rs232-cygwin.h File Reference . . . . .	529
11.63cpp.desire.final/sdh/rs232-vcc.cpp File Reference . . . . .	531
11.64sdh/rs232-vcc.cpp File Reference . . . . .	533
11.65cpp.desire.final/sdh/rs232-vcc.h File Reference . . . . .	535
11.66sdh/rs232-vcc.h File Reference . . . . .	537
11.67cpp.desire.final/sdh/sdh.cpp File Reference . . . . .	539
11.68sdh/sdh.cpp File Reference . . . . .	540
11.69cpp.desire.final/sdh/sdh.h File Reference . . . . .	542
11.70sdh/sdh.h File Reference . . . . .	544
11.71cpp.desire.final/sdh/sdhbase.cpp File Reference . . . . .	546

11.72sdh/sdhbase.cpp File Reference . . . . .	547
11.73cpp.desire.final/sdh/sdhbase.h File Reference . . . . .	548
11.74sdh/sdhbase.h File Reference . . . . .	550
11.75cpp.desire.final/sdh/sdhexception.cpp File Reference . . . . .	552
11.76sdh/sdhexception.cpp File Reference . . . . .	553
11.77cpp.desire.final/sdh/sdhexception.h File Reference . . . . .	554
11.78sdh/sdhexception.h File Reference . . . . .	556
11.79cpp.desire.final/sdh/sdhlibrary_settings.h File Reference . . . . .	558
11.80sdh/sdhlibrary_settings.h File Reference . . . . .	559
11.81cpp.desire.final/sdh/sdhserial.cpp File Reference . . . . .	560
11.82sdh/sdhserial.cpp File Reference . . . . .	561
11.83cpp.desire.final/sdh/sdhserial.h File Reference . . . . .	562
11.84sdh/sdhserial.h File Reference . . . . .	564
11.85cpp.desire.final/sdh/serialbase.cpp File Reference . . . . .	566
11.86sdh/serialbase.cpp File Reference . . . . .	567
11.87cpp.desire.final/sdh/serialbase.h File Reference . . . . .	568
11.88sdh/serialbase.h File Reference . . . . .	570
11.89cpp.desire.final/sdh/simplestringlist.cpp File Reference . . . . .	572
11.90sdh/simplestringlist.cpp File Reference . . . . .	573
11.91cpp.desire.final/sdh/simplestringlist.h File Reference . . . . .	574
11.92sdh/simplestringlist.h File Reference . . . . .	576
11.93cpp.desire.final/sdh/simpletime.h File Reference . . . . .	578
11.94sdh/simpletime.h File Reference . . . . .	579
11.95cpp.desire.final/sdh/simplevector.cpp File Reference . . . . .	580
11.96sdh/simplevector.cpp File Reference . . . . .	581
11.97cpp.desire.final/sdh/simplevector.h File Reference . . . . .	582
11.98sdh/simplevector.h File Reference . . . . .	584
11.99cpp.desire.final/sdh/unit_converter.cpp File Reference . . . . .	585
11.100dh/unit_converter.cpp File Reference . . . . .	586
11.101cpp.desire.final/sdh/unit_converter.h File Reference . . . . .	587
11.102sdh/unit_converter.h File Reference . . . . .	589
11.103pp.desire.final/sdh/util.cpp File Reference . . . . .	590
11.104dh/util.cpp File Reference . . . . .	592
11.105pp.desire.final/sdh/util.h File Reference . . . . .	594
11.106dh/util.h File Reference . . . . .	597
11.107pp.desire.final/sdhllibrary_cpp.dox File Reference . . . . .	600

11.10 <del>8</del> dhlibrary_cpp.dox File Reference . . . . .	601
11.10 <del>9</del> pp.desire.final/vcc/getopt.c File Reference . . . . .	602
11.11 <del>0</del> cc/getopt.c File Reference . . . . .	605
11.11 <del>1</del> pp.desire.final/vcc/getopt.h File Reference . . . . .	608
11.11 <del>2</del> cc/getopt.h File Reference . . . . .	610
11.11 <del>3</del> pp.desire.final/vcc/getopt1.c File Reference . . . . .	612
11.11 <del>4</del> cc/getopt1.c File Reference . . . . .	613
11.11 <del>5</del> demo/demo-griphand.cpp File Reference . . . . .	614
11.11 <del>6</del> doc/onlinehelp-demo-contact-grasping.exe.dox File Reference . . . . .	616
11.11 <del>7</del> doc/onlinehelp-demo-dsa.exe.dox File Reference . . . . .	617
11.11 <del>8</del> doc/onlinehelp-demo-dsa.exe.stackdump.dox File Reference . . . . .	618
11.11 <del>9</del> doc/onlinehelp-demo-GetAxisActualAngle.exe.dox File Reference . . . . .	619
11.12 <del>0</del> doc/onlinehelp-demo-GetFingerXYZ.exe.dox File Reference . . . . .	620
11.12 <del>1</del> doc/onlinehelp-demo-griphand.exe.dox File Reference . . . . .	621
11.12 <del>2</del> doc/onlinehelp-demo-mimic.exe.dox File Reference . . . . .	622
11.12 <del>3</del> doc/onlinehelp-demo-mimic.exe.stackdump.dox File Reference . . . . .	623
11.12 <del>4</del> doc/onlinehelp-demo-ref.exe.dox File Reference . . . . .	624
11.12 <del>5</del> doc/onlinehelp-demo-simple-withtiming.exe.dox File Reference . . . . .	625
11.12 <del>6</del> doc/onlinehelp-demo-simple.exe.dox File Reference . . . . .	626
11.12 <del>7</del> doc/onlinehelp-demo-simple2.exe.dox File Reference . . . . .	627
11.12 <del>8</del> doc/onlinehelp-demo-simple3.exe.dox File Reference . . . . .	628
11.12 <del>9</del> doc/onlinehelp-demo-temperature.exe.dox File Reference . . . . .	629
11.13 <del>0</del> doc/onlinehelp-demo-test.exe.dox File Reference . . . . .	630
11.13 <del>1</del> doc/onlinehelp-demo-velocity-acceleration.exe.dox File Reference . . . . .	631
11.13 <del>2</del> Doxyfile File Reference . . . . .	632
11.13 <del>3</del> Makefile File Reference . . . . .	633

# 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/cscc687y\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/cscc687y(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 <a href="http://www.gnu.org/software/make/manual/make.html"  
<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. The documentation can be found: On the web: <http://www.stack.nl/~dimitri/doxygen/>

## **Chapter 2**

### **Bug List**

**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.

**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.

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

**Member SDH::cSDH::Stop(void)** 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**

**Member SDH::cSDH::WaitAxis(std::vector< int > const &axes, double timeout=-1.0)** 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::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::GripHand(eGraspId grip, double close, double velocity, bool sequ=true)** 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**

**Member SDH::cSDH::GripHand(eGraspId grip, double close, double velocity, bool sequ=true)** 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::EmergencyStop(void)** For now this will **NOT** work while a GripHand() command is executing, even if that was initiated non-sequentially!

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

**Member SDH::cSDH::Stop(void)** 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**

**Member SDH::cSDH::WaitAxis(std::vector< int >const &axes, double timeout=-1.0)** 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::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::GripHand(eGraspId grip, double close, double velocity, bool sequ=true)**  
With SDH firmware > 0.0.2.6 and SDHLlibrary < 0.0.1.12 GripHand() does not work ([Bug 575](#))  
=> Resolved in SDHLlibrary 0.0.1.12

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

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

**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.

**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.



# 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 . . . . .	25
Online help of demonstration programs . . . . .	27



# **Chapter 4**

## **Namespace Index**

### **4.1 Namespace List**

Here is a list of all namespaces with brief descriptions:

<a href="#">SDH</a>	.....	43
---------------------	-------	----



# Chapter 5

## Class Index

### 5.1 Class Hierarchy

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

SDH::cCRC . . . . .	76
SDH::cCRC_DSACON32m . . . . .	80
SDH::cCRC_DSACON32m . . . . .	80
SDH::cDBG . . . . .	83
SDH::cDSA . . . . .	87
SDH::cDSA::sContactInfo . . . . .	104
SDH::cDSA::sControllerInfo . . . . .	105
SDH::cDSA::sMatrixInfo . . . . .	106
SDH::cDSA::sResponse . . . . .	108
SDH::cDSA::sSensorInfo . . . . .	110
SDH::cDSA::sTactileSensorFrame . . . . .	111
SDH::cDSAUpdater . . . . .	116
SDH::cIsGraspedBase . . . . .	120
SDH::cIsGraspedByArea . . . . .	124
SDH::cIsGraspedByArea . . . . .	124
SDH::cMsg . . . . .	128
SDH::cSDHBase . . . . .	268
SDH::cSDH . . . . .	144
SDH::cSDH . . . . .	144
SDH::cSDHSerial . . . . .	306
SDH::cSDHSerial . . . . .	306
SDH::cSDHLibraryException . . . . .	296
SDH::cDSAEception . . . . .	113
SDH::cDSAEception . . . . .	113
SDH::cSDHErrorCommunication . . . . .	290
SDH::cSerialBaseException . . . . .	342
SDH::cCANSerial_ESDEception . . . . .	62
SDH::cCANSerial_ESDEception . . . . .	62
SDH::cCANSerial_PEAKException . . . . .	73
SDH::cCANSerial_PEAKException . . . . .	73
SDH::cRS232Exception . . . . .	141
SDH::cRS232Exception . . . . .	141

SDH::cRS232Exception . . . . .	141
SDH::cRS232Exception . . . . .	141
SDH::cSerialBaseException . . . . .	342
SDH::cSDHErrorCommunication . . . . .	290
SDH::cSDHErrorInvalidParameter . . . . .	293
SDH::cSDHErrorInvalidParameter . . . . .	293
SDH::cSimpleVectorException . . . . .	357
SDH::cSimpleVectorException . . . . .	357
cSDHOptions . . . . .	300
SDH::cSerialBase . . . . .	333
SDH::cCANSerial_ESD . . . . .	53
SDH::cCANSerial_ESD . . . . .	53
SDH::cCANSerial_PEAk . . . . .	65
SDH::cCANSerial_PEAk . . . . .	65
SDH::cRS232 . . . . .	131
SDH::cRS232 . . . . .	131
SDH::cRS232 . . . . .	131
SDH::cRS232 . . . . .	131
SDH::cSimpleStringList . . . . .	345
SDH::cSimpleTime . . . . .	349
SDH::cSimpleVector . . . . .	352
SDH::cUnitConverter . . . . .	360
option . . . . .	367

# Chapter 6

## Class Index

### 6.1 Class List

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

<code>SDH::cCANSerial_ESD</code> (Low-level communication class to access a CAN port ) . . . . .	53
<code>SDH::cCANSerial_ESDEException</code> (Derived exception class for low-level CAN ESD related exceptions ) . . . . .	62
<code>SDH::cCANSerial_PEAK</code> (Low-level communication class to access a CAN port ) . . . . .	65
<code>SDH::cCANSerial_PEAKException</code> (Derived exception class for low-level CAN PEAK related exceptions ) . . . . .	73
<code>SDH::cCRC</code> . . . . .	76
<code>SDH::cCRC_DSACON32m</code> (A derived CRC class that uses a CRC table and initial value suitable for the Weiss Robotics DSACON32m controller) . . . . .	80
<code>SDH::cDBG</code> (A class to print colored debug messages ) . . . . .	83
<code>SDH::cDSA</code> ( <code>SDH::cDSA</code> is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH ) . . . . .	87
<code>SDH::cDSA::sContactInfo</code> (Structure to hold info about the contact of one sensor patch ) . . . . .	104
<code>SDH::cDSA::sControllerInfo</code> (A data structure describing the controller info about the remote DSACON32m controller ) . . . . .	105
<code>SDH::cDSA::sMatrixInfo</code> (A data structure describing a single sensor matrix connected to the remote DSACON32m controller ) . . . . .	106
<code>SDH::cDSA::sResponse</code> (Data structure for storing responses from the remote DSACON32m controller ) . . . . .	108
<code>SDH::cDSA::sSensorInfo</code> (A data structure describing the sensor info about the remote DSACON32m controller ) . . . . .	110
<code>SDH::cDSA::sTactileSensorFrame</code> . . . . .	111
<code>SDH::cDSAException</code> (Derived exception class for low-level DSA related exceptions ) . . . . .	113
<code>SDH::cDSAUpdater</code> . . . . .	116
<code>SDH::cIsGraspedBase</code> . . . . .	120
<code>SDH::cIsGraspedByArea</code> . . . . .	124
<code>SDH::cMsg</code> (Class for short, fixed maximum length text messages ) . . . . .	128
<code>SDH::cRS232</code> (Low-level communication class to access a serial port on Cygwin and Linux ) . .	131
<code>SDH::cRS232Exception</code> (Derived exception class for low-level RS232 related exceptions ) . .	141
<code>SDH::cSDH</code> ( <code>SDH::cSDH</code> is the end user interface class to control a SDH (SCHUNK Dexterous Hand)) . . . . .	144
<code>SDH::cSDHBase</code> (The base class to control the SCHUNK Dexterous Hand ) . . . . .	268

<a href="#">SDH::cSDHErrorCommunication</a> (Derived exception class for exceptions related to communication between the SDHLibrary and the SDH ) . . . . .	290
<a href="#">SDH::cSDHErrorInvalidParameter</a> (Derived exception class for exceptions related to invalid parameters ) . . . . .	293
<a href="#">SDH::cSDHLibraryException</a> (Base class for exceptions in the SDHLibrary-CPP ) . . . . .	296
<a href="#">cSDHOptions</a> . . . . .	300
<a href="#">SDH::cSDHSerial</a> (The class to communicate with a SDH via RS232 ) . . . . .	306
<a href="#">SDH::cSerialBase</a> (Low-level communication class to access a serial port ) . . . . .	333
<a href="#">SDH::cSerialBaseException</a> (Derived exception class for low-level serial communication related exceptions ) . . . . .	342
<a href="#">SDH::cSimpleStringList</a> (A simple string list. (Fixed maximum number of strings of fixed maximum length) ) . . . . .	345
<a href="#">SDH::cSimpleTime</a> (Very simple class to measure elapsed time ) . . . . .	349
<a href="#">SDH::cSimpleVector</a> (A simple vector implementation ) . . . . .	352
<a href="#">SDH::cSimpleVectorException</a> (Derived exception class for low-level simple vector related exceptions ) . . . . .	357
<a href="#">SDH::cUnitConverter</a> (Unit conversion class to convert values between physical unit systems ) . .	360
<a href="#">option</a> . . . . .	367

# Chapter 7

## File Index

### 7.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Doxyfile</a> (Doxyfile for generating documentation for SDHLibrary.cpp using doxygen) . . . . .	632
<a href="#">Makefile</a> (Makefile for SDH SDHLibrary C project) . . . . .	633
cpp.desire.final/demo/ <a href="#">canact.cpp</a> (Yet incomplete tool to send and receive data via CAN. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	383
cpp.desire.final/demo/ <a href="#">demo-contact-grasping.cpp</a> (Simple script to do grasping using tactile sensor info feedback. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	387
cpp.desire.final/demo/ <a href="#">demo-dsa.cpp</a> (Simple program to test class cDSA. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	393
cpp.desire.final/demo/ <a href="#">demo-GetAxisActualAngle.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	399
cpp.desire.final/demo/ <a href="#">demo-GetFingerXYZ.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	405
cpp.desire.final/demo/ <a href="#">demo-mimic.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	411
cpp.desire.final/demo/ <a href="#">demo-simple-withtiming.cpp</a> (Very simple C++ programm to make an attached SDH move) . . . . .	415
cpp.desire.final/demo/ <a href="#">demo-simple.cpp</a> (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	419
cpp.desire.final/demo/ <a href="#">demo-simple2.cpp</a> (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Stop. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	423
cpp.desire.final/demo/ <a href="#">demo-simple3.cpp</a> (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and WaitAxis. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	427
cpp.desire.final/demo/ <a href="#">demo-temperature.cpp</a> (Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options) . . . . .	431

cpp.desire.final/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 <a href="#">_help_</a> and online help ("h" or "-help") for available options) . . . . .	435
cpp.desire.final/demo/dsaboost.cpp (Helper stuff for the "boosted" DSA stuff) . . . . .	439
cpp.desire.final/demo/dsaboost.h (Helper stuff for the DSA using boost) . . . . .	443
cpp.desire.final/demo/sdhoptions.cpp (Implementation of a class to parse common SDH related command line options) . . . . .	447
cpp.desire.final/demo/sdhoptions.h (Implementation of a class to parse common SDH related command line options) . . . . .	459
cpp.desire.final/sdh/basicdef.h (This file contains some basic definitions (defines, macros, datatypes)) . . . . .	463
cpp.desire.final/sdh/canserial-esd.cpp (Implementation of class <a href="#">SDH::cCANSerial_ESD</a> , a class to access an ESD CAN interface on cygwin/linux and Visual Studio) . . . . .	467
cpp.desire.final/sdh/canserial-esd.h (Interface of class <a href="#">SDH::cCANSerial_ESD</a> , class to access CAN bus via ESD card on cygwin/linux) . . . . .	471
cpp.desire.final/sdh/canserial-peak.cpp (Implementation of class <a href="#">SDH::cCANSerial_PEAK</a> , a class to access a PEAK CAN interface on cygwin/linux and Visual Studio) . . . . .	475
cpp.desire.final/sdh/canserial-peak.h (Interface of class <a href="#">SDH::cCANSerial_PEAK</a> , class to access CAN bus via PEAK card on cygwin/linux) . . . . .	479
cpp.desire.final/sdh/crc.cpp (Implementation of class <a href="#">SDH::cCRC_DSACON32m</a> (actually only the static members all other is derived)) . . . . .	483
cpp.desire.final/sdh/crc.h (This file contains interface to cCRC, a class to handle CRC calculation)	485
cpp.desire.final/sdh/dbg.h (This file contains interface and implementation of class <a href="#">SDH::cDBG</a> , a class for colorfull debug messages) . . . . .	489
cpp.desire.final/sdh/dsa.cpp (This file contains definition of <a href="#">SDH::cDSA</a> , a class to communicate with the tactile sensors of the SDH) . . . . .	493
cpp.desire.final/sdh/dsa.h (This file contains interface to <a href="#">SDH::cDSA</a> , a class to communicate with the tactile sensors of the SDH) . . . . .	497
cpp.desire.final/sdh/release.h (This file contains nothing but C/C++ defines with the name of the project itself ( <a href="#">PROJECT_NAME</a> ) and the name of the release ( <a href="#">PROJECT_RELEASE</a> ) of the whole project) . . . . .	503
cpp.desire.final/sdh/rs232-cygwin.cpp (Implementation of class <a href="#">SDH::cRS232</a> , a class to access serial RS232 port on cygwin/linux) . . . . .	523
cpp.desire.final/sdh/rs232-cygwin.h (Interface of class <a href="#">SDH::cRS232</a> , a class to access serial RS232 port on cygwin/linux) . . . . .	527
cpp.desire.final/sdh/rs232-vcc.cpp (Implementation of class <a href="#">SDH::cRS232</a> , a class to access serial RS232 port with VCC compiler on Windows) . . . . .	531
cpp.desire.final/sdh/rs232-vcc.h (Implementation of class <a href="#">SDH::cRS232</a> , a class to access serial RS232 port with VCC compiler on Windows) . . . . .	535
cpp.desire.final/sdh/sdh.cpp (This file contains the interface to class <a href="#">SDH::cSDH</a> , the end user class to access the SDH from a PC) . . . . .	539
cpp.desire.final/sdh/sdh.h (This file contains the interface to class <a href="#">SDH::cSDH</a> , the end user class to access the SDH from a PC) . . . . .	542
cpp.desire.final/sdh/sdhbase.cpp (Implementation of class <a href="#">SDH::cSDHBase</a> ) . . . . .	546
cpp.desire.final/sdh/sdhbase.h (Interface of class <a href="#">SDH::cSDHBase</a> ) . . . . .	548
cpp.desire.final/sdh/sdhexception.cpp (Implementation of the exception base class <a href="#">SDH::cSDHLibraryException</a> and <a href="#">SDH::cMsg</a> ) . . . . .	552
cpp.desire.final/sdh/sdhexception.h (Interface of the exception base class <a href="#">SDH::cSDHLibraryException</a> and <a href="#">SDH::cMsg</a> ) . . . . .	554
cpp.desire.final/sdh/sdhlibrary_settings.h (This file contains settings to make the SDHLibrary compile on different systems:	
• gcc/Cygwin/Windows	

• gcc/Linux	
• VisualC++/Windows	
)	558
cpp.desire.final/sdh/ <a href="#">sdhserial.cpp</a> (Interface of class <code>SDH::cSDHSerial</code> )	560
cpp.desire.final/sdh/ <a href="#">sdhserial.h</a> (Interface of class <code>SDH::cSDHSerial</code> )	562
cpp.desire.final/sdh/ <a href="#">serialbase.cpp</a> (Implementation of class <code>SDH::cSerialBase</code> , a virtual base class to access serial interfaces like RS232 or CAN)	566
cpp.desire.final/sdh/ <a href="#">serialbase.h</a> (Interface of class <code>SDH::cSerialBase</code> , a virtual base class to access serial communication channels like RS232 or CAN)	568
cpp.desire.final/sdh/ <a href="#">simplestringlist.cpp</a> (Implementation of class <code>SDH::cSimpleStringList</code> )	572
cpp.desire.final/sdh/ <a href="#">simplestringlist.h</a> (Interface of class <code>SDH::cSimpleStringList</code> )	574
cpp.desire.final/sdh/ <a href="#">simpletime.h</a> (Interface of auxilliary utility functions for SDHLibrary-CPP)	578
cpp.desire.final/sdh/ <a href="#">simplevector.cpp</a> (Implementation of class <code>SDH::cSimpleVector</code> )	580
cpp.desire.final/sdh/ <a href="#">simplevector.h</a> (Interface of class <code>SDH::cSimpleVector</code> )	582
cpp.desire.final/sdh/ <a href="#">unit_converter.cpp</a> (Implementation of class <code>SDH::cUnitConverter</code> )	585
cpp.desire.final/sdh/ <a href="#">unit_converter.h</a> (Interface of class <code>SDH::cUnitConverter</code> )	587
cpp.desire.final/sdh/ <a href="#">util.cpp</a> (Implementation of auxilliary utility functions for SDHLibrary-CPP)	590
cpp.desire.final/sdh/ <a href="#">util.h</a> (Interface of auxilliary utility functions for SDHLibrary-CPP)	594
cpp.desire.final/vcc/ <a href="#">getopt.c</a>	602
cpp.desire.final/vcc/ <a href="#">getopt.h</a>	608
cpp.desire.final/vcc/ <a href="#">getopt1.c</a>	612
demo/ <a href="#">cancat.cpp</a> (Yet incomplete tool to send and receive data via CAN. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	385
demo/ <a href="#">demo-contact-grasping.cpp</a> (Simple script to do grasping using tactile sensor info feedback. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	390
demo/ <a href="#">demo-dsa.cpp</a> (Simple program to test class cDSA. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	396
demo/ <a href="#">demo-GetAxisActualAngle.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	402
demo/ <a href="#">demo-GetFingerXYZ.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	408
demo/ <a href="#">demo-griphand.cpp</a> (Very simple demonstration program using the SDHLibrary-CPP: Demonstrate the use of the GripHand command See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	614
demo/ <a href="#">demo-mimic.cpp</a> (Print measured actual axis angles of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	413
demo/ <a href="#">demo-simple-withtiming.cpp</a> (Very simple C++ programm to make an attached SDH move)	417
demo/ <a href="#">demo-simple.cpp</a> (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger with "pose" controller type (coordinated position control). See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	421
demo/ <a href="#">demo-simple2.cpp</a> (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Stop. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	425
demo/ <a href="#">demo-simple3.cpp</a> (Very simple C++ programm to make an attached SDH move. With non-sequential call of move and WaitAxis. See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	429
demo/ <a href="#">demo-temperature.cpp</a> (Print measured temperatures of an attached SDH. (C++ demo application using the SDHLibrary-CPP library.) See <a href="#">_help_</a> and online help (" -h " or " -help ") for available options)	433

<code>demo/demo-velocity-acceleration.cpp</code> (Very simple demonstration program using the SDHLibrary-CPP: Make an attached SDH move one finger in "velocity with acceleration ramp" control mode. See <code>_help_</code> and online help (" <code>-h</code> " or " <code>--help</code> ") for available options ) . . . . .	437
<code>demo/dsaboost.cpp</code> (Helper stuff for the "boosted" DSA stuff ) . . . . .	441
<code>demo/dsaboost.h</code> (Helper stuff for the DSA using boost ) . . . . .	445
<code>demo/sdhoptions.cpp</code> (Implementation of a class to parse common SDH related command line options ) . . . . .	453
<code>demo/sdhoptions.h</code> (Implementation of a class to parse common SDH related command line options ) . . . . .	461
<code>sdh/basisdef.h</code> (This file contains some basic definitions (defines, macros, datatypes) ) . . . . .	465
<code>sdh/canserial-esd.cpp</code> (Implementation of class <code>SDH::cCANSerial_ESD</code> , a class to access an ESD CAN interface on cygwin/linux and Visual Studio ) . . . . .	469
<code>sdh/canserial-esd.h</code> (Interface of class <code>SDH::cCANSerial_ESD</code> , class to access CAN bus via ESD card on cygwin/linux ) . . . . .	473
<code>sdh/canserial-peak.cpp</code> (Implementation of class <code>SDH::cCANSerial_PEAK</code> , a class to access a PEAK CAN interface on cygwin/linux and Visual Studio ) . . . . .	477
<code>sdh/canserial-peak.h</code> (Interface of class <code>SDH::cCANSerial_PEAK</code> , class to access CAN bus via PEAK card on cygwin/linux ) . . . . .	481
<code>sdh/crc.cpp</code> (Implementation of class <code>SDH::cCRC_DSACON32m</code> (actually only the static members all other is derived) ) . . . . .	484
<code>sdh/crc.h</code> (This file contains interface to cCRC, a class to handle CRC calculation ) . . . . .	487
<code>sdh/dbg.h</code> (This file contains interface and implementation of class <code>SDH::cDBG</code> , a class for col-orfull debug messages ) . . . . .	491
<code>sdh/dsa.cpp</code> (This file contains definition of <code>SDH::cDSA</code> , a class to communicate with the tactile sensors of the SDH ) . . . . .	495
<code>sdh/dsa.h</code> (This file contains interface to <code>SDH::cDSA</code> , a class to communicate with the tactile sensors of the SDH ) . . . . .	500
<code>sdh/release.h</code> (This file contains nothing but C/C++ defines with the name of the project itself ( <code>PROJECT_NAME</code> ) and the name of the release ( <code>PROJECT_RELEASE</code> ) of the whole project ) . . . . .	513
<code>sdh/rs232-cygwin.cpp</code> (Implementation of class <code>SDH::cRS232</code> , a class to access serial RS232 port on cygwin/linux ) . . . . .	525
<code>sdh/rs232-cygwin.h</code> (Interface of class <code>SDH::cRS232</code> , a class to access serial RS232 port on cygwin/linux ) . . . . .	529
<code>sdh/rs232-vcc.cpp</code> (Implementation of class <code>SDH::cRS232</code> , a class to access serial RS232 port with VCC compiler on Windows ) . . . . .	533
<code>sdh/rs232-vcc.h</code> (Implementation of class <code>SDH::cRS232</code> , a class to access serial RS232 port with VCC compiler on Windows ) . . . . .	537
<code>sdh/sdh.cpp</code> (This file contains the interface to class <code>SDH::cSDH</code> , the end user class to access the SDH from a PC ) . . . . .	540
<code>sdh/sdh.h</code> (This file contains the interface to class <code>SDH::cSDH</code> , the end user class to access the SDH from a PC ) . . . . .	544
<code>sdh/sdhbase.cpp</code> (Implementation of class <code>SDH::cSDHBase</code> ) . . . . .	547
<code>sdh/sdhbase.h</code> (Interface of class <code>SDH::cSDHBase</code> ) . . . . .	550
<code>sdh/sdhexception.cpp</code> (Implementation of the exception base class <code>SDH::cSDHLibraryException</code> and <code>SDH::cMsg</code> ) . . . . .	553
<code>sdh/sdhexception.h</code> (Interface of the exception base class <code>SDH::cSDHLibraryException</code> and <code>SDH::cMsg</code> ) . . . . .	556
<code>sdh/sdhlibrary_settings.h</code> (This file contains settings to make the SDHLibrary compile on different systems:	

- gcc/Cygwin/Windows
- gcc/Linux

• VisualC++/Windows	
)	559
sdh/ <a href="#">sdhserial.cpp</a> (Interface of class <b>SDH::cSDHSerial</b> )	561
sdh/ <a href="#">sdhserial.h</a> (Interface of class <b>SDH::cSDHSerial</b> )	564
sdh/ <a href="#">serialbase.cpp</a> (Implementation of class <b>SDH::cSerialBase</b> , a virtual base class to access serial interfaces like RS232 or CAN )	567
sdh/ <a href="#">serialbase.h</a> (Interface of class <b>SDH::cSerialBase</b> , a virtual base class to access serial communication channels like RS232 or CAN )	570
sdh/ <a href="#">simplestringlist.cpp</a> (Implementation of class <b>SDH::cSimpleStringList</b> )	573
sdh/ <a href="#">simplestringlist.h</a> (Interface of class <b>SDH::cSimpleStringList</b> )	576
sdh/ <a href="#">simpletime.h</a> (Interface of auxilliary utility functions for SDHLibrary-CPP)	579
sdh/ <a href="#">simplevector.cpp</a> (Implementation of class <b>SDH::cSimpleVector</b> )	581
sdh/ <a href="#">simplevector.h</a> (Interface of class <b>SDH::cSimpleVector</b> )	584
sdh/ <a href="#">unit_converter.cpp</a> (Implementation of class <b>SDH::cUnitConverter</b> )	586
sdh/ <a href="#">unit_converter.h</a> (Interface of class <b>SDH::cUnitConverter</b> )	589
sdh/ <a href="#">util.cpp</a> (Implementation of auxilliary utility functions for SDHLibrary-CPP)	592
sdh/ <a href="#">util.h</a> (Interface of auxilliary utility functions for SDHLibrary-CPP)	597
vcc/ <a href="#">getopt.c</a>	605
vcc/ <a href="#">getopt.h</a>	610
vcc/ <a href="#">getopt1.c</a>	613



# Chapter 8

## Module Documentation

### 8.1 Compile time settings

#### 8.1.1 Detailed Description

Primary settings to be adjusted by the user.

#### 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_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.*

#### 8.1.2 Define Documentation

##### 8.1.2.1 `#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.

##### 8.1.2.2 `#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

### 8.2.1 Detailed Description

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

#### Defines

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

### 8.2.2 Define Documentation

8.2.2.1 `#define NAMESPACE_SDH_END }`

8.2.2.2 `#define NAMESPACE_SDH_END }`

8.2.2.3 `#define NAMESPACE_SDH_START namespace SDH {`

8.2.2.4 `#define NAMESPACE_SDH_START namespace SDH {`

8.2.2.5 `#define USING_NAMESPACE_SDH using namespace SDH;`

8.2.2.6 `#define USING_NAMESPACE_SDH using namespace SDH;`

## 8.3 Demonstration programs

### 8.3.1 Detailed Description

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

#### Files

- 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.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-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-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++ 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.*

- file [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.*

- 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.*

- 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.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-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++ 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.*

- file [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.*

- 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.*

## 8.4 Online help of demonstration programs

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

### 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 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/ttyUSB0
  - 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.

-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).

DSA options (tactile sensor):
-q PORT, --dsaport=PORT
    use RS232 communication PORT to connect to tactile sensor controller
    of SDH instead of default 1='COM2'='/dev/ttys1'.

--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/ttysUSB%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.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 full frames continuously from tactile sensors connected to port 3 = COM4:
 

```
> demo-dsa --dsaport=3 -f -r 1
```
  - Read a single full frame from tactile sensors connected to USB to RS232 converter 0:
 

```
> demo-dsa --dsa_rs_device=/dev/ttysUSB0 -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 to
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)

```
> demo-dsa -p 2 --dsaport=3 -v
```

- 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.

-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 1='COM2'='/dev/ttys1'.

--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/ttysUSB%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.
```

### 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 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 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.

-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) .

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

-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 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/ttyUSBO
  - 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.

-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).

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

-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 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 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.

-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).

```

### 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:
- Print actual angles of an SDH connected to port 2 = COM3 once:  
> demo-mimic -p 2
- Print actual angles of an SDH connected to port 2 = COM3 every 500ms:  
> demo-mimic -p 2 -t 0.5
- Print actual angles of an SDH connected to USB to RS232 converter 0 once:  
> demo-mimic --sdh\_rs\_device=/dev/ttyUSB0
- Get the version info of an SDH connected to port 2 = COM3

```

> demo-mimic --port=2 -v

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.

-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).

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

-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-ref.exe

### 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.

-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).
```

## 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 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 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.

-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).
```

### 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 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 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.  
  
-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) .
```

### 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 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 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.

-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).
```

### 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 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 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.

-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).

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

-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.

-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).

```

### 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.)

```

- 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 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.

-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).
```

# Chapter 9

## Namespace Documentation

### 9.1 SDH Namespace Reference

#### 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.

#### Classes

- class `cDSAUpdater`
- class `cIsGraspedBase`
- class `cIsGraspedByArea`
- 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.*

- 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.*

- class `cCRC`

- class `cCRC_DSACON32m`

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

- class `cDBG`

*A class to print colored debug messages.*

- class `cDSAEexception`

*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.*

- 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 **cUnitConverter**

*Unit conversion class to convert values between physical unit systems.*

## 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 **UInt16 tCRCValue**

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

- typedef void \* **NTCAN\_HANDLE**

*dummy definition in case ntcan.h is not available*

- 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 double(**cUnitConverter::\* pDoubleUnitConverterFunction** )(double) const

*Type of a pointer to a function like 'double cUnitConverter::ToExternal( double )' or 'cUnitConverterToInternal( double )'.*

- typedef void \* **PCAN\_HANDLE**

*dummy definition in case Pcan\_usb.h is not available*

## Functions

- std::ostream & `operator<<` (std::ostream &stream, `cDSA::sControllerInfo` const &controller\_info)
- std::ostream & `operator<<` (std::ostream &stream, `cDSA::sSensorInfo` const &sensor\_info)
- std::ostream & `operator<<` (std::ostream &stream, `cDSA::sMatrixInfo` const &matrix\_info)
- std::ostream & `operator<<` (std::ostream &stream, `cDSA::sResponse` const &response)
- std::ostream & `operator<<` (std::ostream &stream, `cDSA` const &dsa)
- std::ostream & `operator<<` (std::ostream &stream, `cMsg` const &msg)
- std::ostream & `operator<<` (std::ostream &stream, `cSDHLibraryException` const &e)
- std::ostream & `operator<<` (std::ostream &stream, `cSimpleStringList` const &ssl)

*Output of `cSimpleStringList` objects in 'normal' output streams.*

- std::vector< int > `NumerifyRelease` (char const \*rev)
- `cUnitConverter` const `uc_identity` ("any","any","?", 1.0, 0.0, 4)

## 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

- std::ostream \* `g_sdh_debug_log` = &std::cerr
  - `cUnitConverter` const `uc_identity` ("any","any","?", 1.0, 0.0, 4)
- Identity converter (internal = external).*
- double `M_PI` = 4.0\*atan(1.0)

### 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::NTCAN\_HANDLE**

dummy definition in case ntcan.h is not available

#### 9.1.2.5 **typedef void\* SDH::PCAN\_HANDLE**

dummy definition in case Pcan\_usb.h is not available

#### 9.1.2.6 **typedef double(cUnitConverter::\* SDH::pDoubleUnitConverterFunction)(double) const**

Type of a pointer to a function like 'double cUnitConverter::ToExternal( double )' or 'cUnitConverterToInternal( double )'.

#### 9.1.2.7 **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](#).

#### 9.1.2.8 **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](#).

#### 9.1.2.9 **typedef UInt16 SDH::tCRCValue**

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

#### 9.1.2.10 **typedef uint16\_t SDH::UInt16**

unsigned integer, size 2 Byte (16 Bit)

**9.1.2.11 `typedef uint32_t SDH::UInt32`**

unsigned integer, size 4 Byte (32 Bit)

**9.1.2.12 `typedef uint8_t SDH::UInt8`**

unsigned integer, size 1 Byte (8 Bit)

**9.1.3 Function Documentation****9.1.3.1 `template<typename Function, typename InputIterator> Function SDH::apply (Function f, InputIterator first, InputIterator last) [inline]`**

Apply a function to every element of a sequence.

**Parameters:**

*first* An input iterator.

*last* An input iterator.

*f* A unary function object.

**Returns:**

*f*.

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

**9.1.3.2 `template<typename Function, typename Tp> void SDH::apply (Function f, Tp & sequence) [inline]`**

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

**Parameters:**

*f* A unary function object.

*sequence* The iterable sequence to modify.

Applies the function object *f* to each element of the *sequence*.

**9.1.3.3 `bool SDH::Approx (int n, double * a, double * b, double * eps)`**

Return True if list/tuple/array *a*=(*a*<sub>1</sub>,*a*<sub>2</sub>,...) is approximately the same as *b*=(*b*<sub>1</sub>,*b*<sub>2</sub>,...). I.E. |*a*<sub>i</sub>-*b*<sub>i</sub>| < *eps*[*i*]

**9.1.3.4 `double SDH::Approx (double a, double b, double eps)`**

Return True if *a* is approximately the same as *b*. I.E. |*a*-*b*| < *eps*

**9.1.3.5 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:**

*rev1* - a release string like "0.0.1.5" or "0.0.1.11-a

*rev2* - 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

**9.1.3.6 double SDH::DegToRad (double *d*)**

Return d in deg converted to rad

**9.1.3.7 bool SDH::InIndex (int *v*, int *max*)**

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

**9.1.3.8 bool SDH::InRange (int *n*, double const \* *v*, double const \* *min*, double const \* *max*)**

Return True if in list/tuple/array v=(v1,v2,...) each v\_i is in range [min\_i..max\_i] with min = (min1, min2,...) max = (max1, max2, ...)

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

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

**9.1.3.10 template<typename Function, typename Tp> Tp SDH::map (Function *f*, Tp *sequence* [inline])**

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

**Parameters:**

*f* - A unary function object.

*sequence* - An iterable object.

**Returns:**

copy of *sequence* with the mapped elements

**9.1.3.11 std::vector< int > SDH::NumerifyRelease (char const \* *rev*)**

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

**Parameters:**

*rev* release string like "0.0.1.11-a"

**Returns:**

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

**9.1.3.12 template<typename T> std::ostream & SDH::operator<< (std::ostream & *stream*, std::vector< T > const & *v*) [inline]**

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

**Parameters:**

*stream* - the output stream to insert into

*v* - the vector of objects to insert into *stream*

**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 >>operator<<<< in >>
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
```

---

**9.1.3.13 std::ostream & SDH::operator<< (std::ostream & *stream*, cSimpleStringList const & *ssl*)**

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

**9.1.3.14 std::ostream & SDH::operator<< (std::ostream & *stream*, cSDHLibraryException const & *e*)**

**9.1.3.15 std::ostream & SDH::operator<< (std::ostream & *stream*, cMsg const & *msg*)**

**9.1.3.16 std::ostream & SDH::operator<< (std::ostream & *stream*, cDSA const & *dsa*)**

**9.1.3.17 std::ostream & SDH::operator<< (std::ostream & *stream*, cDSA::sResponse const & *response*)**

**9.1.3.18 std::ostream & SDH::operator<< (std::ostream & *stream*, cDSA::sMatrixInfo const & *matrix\_info*)**

**9.1.3.19 std::ostream & SDH::operator<< (std::ostream & *stream*, cDSA::sSensorInfo const & *sensor\_info*)**

**9.1.3.20 std::ostream & SDH::operator<< (std::ostream & *stream*, cDSA::sControllerInfo const & *controller\_info*)**

**9.1.3.21 double SDH::RadToDeg (double *r*)**

Return r in rad converted to deg

**9.1.3.22 void SDH::SleepSec (double *t*)**

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

**9.1.3.23 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 = (min1, min2,...) max = (max1, max2, ..) This modifies v!

**9.1.3.24 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 = (min1, min2,...) max = (max1, max2, ..) This modifies \*v!

**9.1.3.25 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 = (min1, min2,...) max = (max1, max2, ..) This modifies \*v!

**9.1.3.26 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

**9.1.3.27 cUnitConverter const SDH::uc\_identity ("any", "any", "?", 1.0, 0.0, 4)****9.1.4 Variable Documentation****9.1.4.1 std::ostream \* SDH::g\_sdh\_debug\_log = &std::cerr****9.1.4.2 double SDH::M\_PI = 4.0\*atan(1.0)****9.1.4.3 cUnitConverter const SDH::uc\_identity**

Identity converter (internal = external).

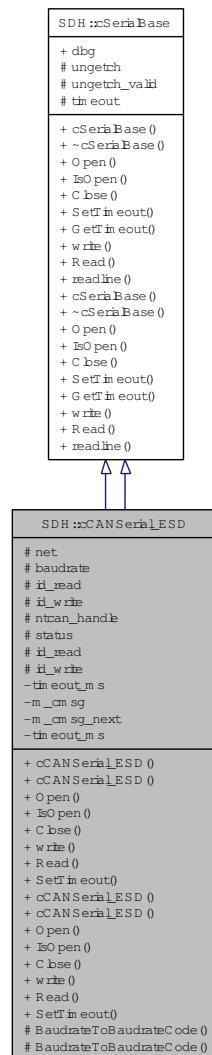
# **Chapter 10**

## **Class Documentation**

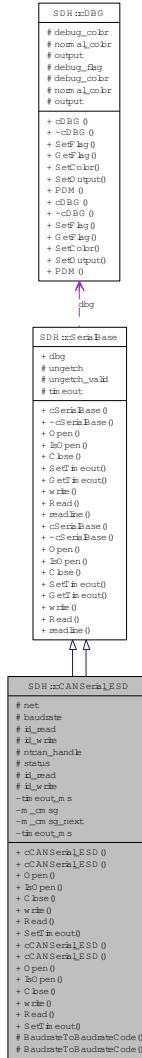
### **10.1 SDH::cCANSerial\_ESD Class Reference**

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

Inheritance diagram for SDH::cCANSerial\_ESD:



Collaboration diagram for SDH::cCANSerial\_ESD:



### 10.1.1 Detailed Description

Low-level communication class to access a CAN port.

#### Public Member Functions

- [cCANSerial\\_ESD](#) (int \_net, unsigned long \_baudrate, double \_timeout, int32\_t \_id\_read, int32\_t \_id\_write) throw (cCANSerial\_ESDEception\*)
- [cCANSerial\\_ESD \(NTCAN\\_HANDLE \\_ntcan\\_handle, double \\_timeout, int32\\_t \\_id\\_read, int32\\_t \\_id\\_write\)](#) throw (cCANSerial\_ESDEception\*)
- void [Open](#) (void) throw (cCANSerial\_ESDEception\*)
- 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\*)
 

*(cCANSerial\_ESDException\*)*
- void **SetTimeout** (double \_timeout) throw (cSerialBaseException\*)
 

*set the timeout for next `readline()` calls (negative value means: no timeout, wait for ever)*
- **cCANSerial\_ESD** (int \_net, unsigned long \_baudrate, double \_timeout, Int32 \_id\_read, Int32 \_id\_write) throw (cCANSerial\_ESDException\*)
 

*(cCANSerial\_ESDException\*)*
- **cCANSerial\_ESD (NTCAN\_HANDLE \_ntcan\_handle, double \_timeout, Int32 \_id\_read, Int32 \_id\_write)** throw (cCANSerial\_ESDException\*)
 

*(cCANSerial\_ESDException\*)*
- void **Open** (void) throw (cCANSerial\_ESDException\*)
 

*(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\*)
 

*(cCANSerial\_ESDException\*)*
- void **SetTimeout** (double \_timeout) throw (cSerialBaseException\*)
 

*set the timeout for next `readline()` calls (negative value means: no timeout, wait for ever)*

## Protected Member Functions

- uint32\_t **BaudrateToBaudrateCode** (unsigned long **baudrate**) throw (cCANSerial\_ESDException\*)
 

*Translate a baudrate given as unsigned long into a baudrate code for struct termios.*
- UInt32 **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*
- int32\_t **id\_read**

*the CAN ID used for reading*

- `int32_t id_write`  
*the CAN ID used for writing*

- `NTCAN_HANDLE ntcan_handle`  
*the handle to the driver*

- `int status`
- `Int32 id_read`  
*the CAN ID used for reading*

- `Int32 id_write`  
*the CAN ID used for writing*

## 10.1.2 Constructor & Destructor Documentation

### 10.1.2.1 cCANSerial\_ESD::cCANSerial\_ESD (`int _net, unsigned long _baudrate, double _timeout, int32_t _id_read, int32_t _id_write`) throw (`cCANSerial_ESDError*`)

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

#### Parameters:

`_net` - the ESD CAN net to use

`_baudrate` - the baudrate in bit/s. Only some bitrates are valid:  
(1000000,800000,500000,250000,125000,100000,50000,20000,10000)

`_timeout` - the timeout in seconds (0 for no timeout = wait for ever)

`_id_read` - the CAN ID to use for reading (The SDH sends data on this ID)

`_id_write` - the CAN ID to use for writing (The SDH receives data on this ID)

### 10.1.2.2 cCANSerial\_ESD::cCANSerial\_ESD (`NTCAN_HANDLE _ntcan_handle, double _timeout, int32_t _id_read, int32_t _id_write`) throw (`cCANSerial_ESDError*`)

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

#### Parameters:

`_ntcan_handle` - the ESD CAN handle to reuse

`_timeout` - the timeout in seconds (0 for no timeout = wait for ever)

`_id_read` - the CAN ID to use for reading (The SDH sends data on this ID)

`_id_write` - the CAN ID to use for writing (The SDH receives data on this ID)

### 10.1.2.3 SDH::cCANSerial\_ESD::cCANSerial\_ESD (int \_net, unsigned long \_baudrate, double \_timeout, Int32 \_id\_read, Int32 \_id\_write) throw (cCANSerial\_ESDEception\*)

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

**Parameters:**

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

### 10.1.2.4 SDH::cCANSerial\_ESD::cCANSerial\_ESD (NTCAN\_HANDLE \_ntcan\_handle, double \_timeout, Int32 \_id\_read, Int32 \_id\_write) throw (cCANSerial\_ESDEception\*)

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

**Parameters:**

- \_ntcan\_handle* - the ESD CAN handle to reuse
- \_timeout* - the timeout in seconds (0 for no timeout = wait for ever)
- \_id\_read* - the CAN ID to use for reading (The SDH sends data on this ID)
- \_id\_write* - the CAN ID to use for writing (The SDH receives data on this ID)

## 10.1.3 Member Function Documentation

### 10.1.3.1 UInt32 cCANSerial\_ESD::BaudrateToBaudrateCode (unsigned long *baudrate*) throw (cCANSerial\_ESDEception\*) [protected]

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

### 10.1.3.2 void cCANSerial\_ESD::Open (void) throw (cCANSerial\_ESDEception\*) [virtual]

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

Implements [SDH::cSerialBase](#).

### 10.1.3.3 bool cCANSerial\_ESD::IsOpen (void) throw () [virtual]

Return true if interface to CAN ESD is open.

Implements [SDH::cSerialBase](#).

### 10.1.3.4 void cCANSerial\_ESD::Close (void) throw (cCANSerial\_ESDEception\*) [virtual]

Close the previously opened CAN ESD interface port.

Implements [SDH::cSerialBase](#).

**10.1.3.5 int cCANSerial\_ESD::write (char const \* *ptr*, int *len* = 0) throw (cCANSerial\_ESDError\*) [virtual]**

Write data to a previously opened port.

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

**Parameters:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

**10.1.3.6 ssize\_t cCANSerial\_ESD::Read (void \* *data*, ssize\_t *size*, long *timeout\_us*, bool *return\_on\_less\_data*) throw (cCANSerial\_ESDError\*) [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](#).

**10.1.3.7 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](#).

**10.1.3.8 UInt32 SDH::cCANSerial\_ESD::BaudrateToBaudrateCode (unsigned long *baudrate*) throw (cCANSerial\_ESDError\*) [protected]**

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

**10.1.3.9 void SDH::cCANSerial\_ESD::Open (void) throw (cCANSerial\_ESDError\*) [virtual]**

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

Implements [SDH::cSerialBase](#).

**10.1.3.10 bool SDH::cCANSerial\_ESD::IsOpen (void) throw () [virtual]**

Return true if interface to CAN ESD is open.

Implements [SDH::cSerialBase](#).

---

**10.1.3.11 void SDH::cCANSerial\_ESD::Close (void) throw (cCANSerial\_ESDEception\*) [virtual]**

Close the previously opened CAN ESD interface port.

Implements [SDH::cSerialBase](#).

**10.1.3.12 int SDH::cCANSerial\_ESD::write (char const \* *ptr*, int *len* = 0) throw (cCANSerial\_ESDEception\*) [virtual]**

Write data to a previously opened port.

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

**Parameters:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

**10.1.3.13 ssize\_t SDH::cCANSerial\_ESD::Read (void \* *data*, ssize\_t *size*, long *timeout\_us*, bool *return\_on\_less\_data*) throw (cCANSerial\_ESDEception\*) [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](#).

**10.1.3.14 void SDH::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](#).

## 10.1.4 Member Data Documentation

**10.1.4.1 int SDH::cCANSerial\_ESD::net [protected]**

the ESD CAN net to use

**10.1.4.2 unsigned long SDH::cCANSerial\_ESD::baudrate [protected]**

the baudrate to use in bit/s

**10.1.4.3 int32\_t SDH::cCANSerial\_ESD::id\_read [protected]**

the CAN ID used for reading

**10.1.4.4 int32\_t SDH::cCANSerial\_ESD::id\_write [protected]**

the CAN ID used for writing

**10.1.4.5 NTCAN\_HANDLE SDH::cCANSerial\_ESD::ntcan\_handle [protected]**

the handle to the driver

**10.1.4.6 int SDH::cCANSerial\_ESD::status [protected]****10.1.4.7 Int32 SDH::cCANSerial\_ESD::id\_read [protected]**

the CAN ID used for reading

**10.1.4.8 Int32 SDH::cCANSerial\_ESD::id\_write [protected]**

the CAN ID used for writing

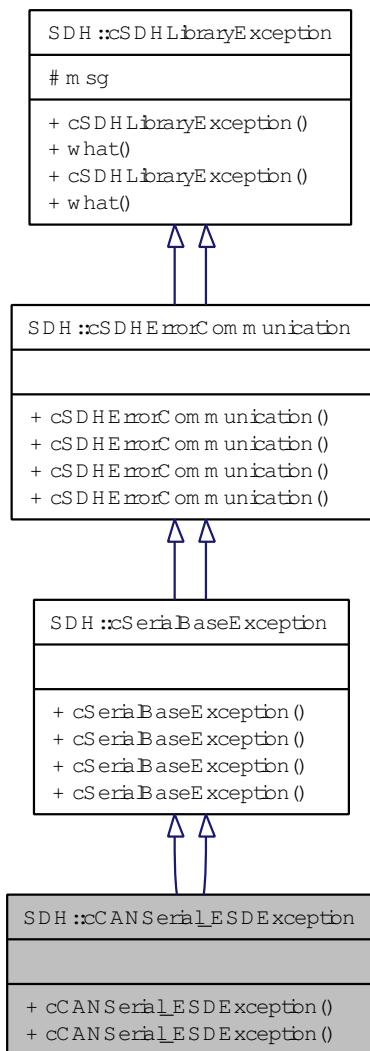
The documentation for this class was generated from the following files:

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

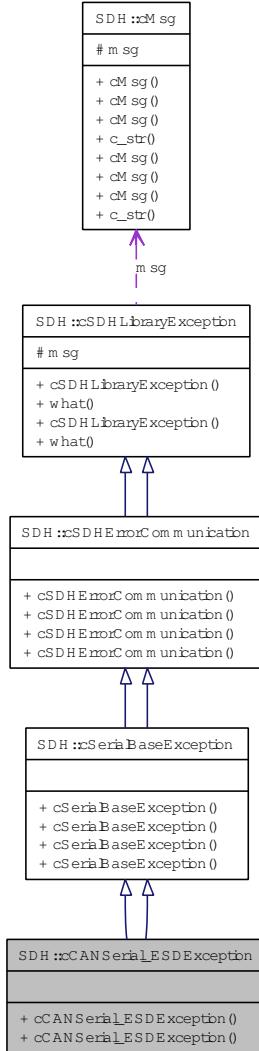
## 10.2 SDH::cCANSerial\_ESDEception Class Reference

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

Inheritance diagram for SDH::cCANSerial\_ESDEception:



Collaboration diagram for SDH::cCANSerial\_ESDEception:



### 10.2.1 Detailed Description

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

### Public Member Functions

- [cCANSerial\\_ESDEception \(cMsg const &\\_msg\)](#)
- [cCANSerial\\_ESDEception \(cMsg const &\\_msg\)](#)

### 10.2.2 Constructor & Destructor Documentation

**10.2.2.1 SDH::cCANSerial\_ESDException::cCANSerial\_ESDException (cMsg const & \_msg)**  
[inline]

**10.2.2.2 SDH::cCANSerial\_ESDException::cCANSerial\_ESDException (cMsg const & \_msg)**  
[inline]

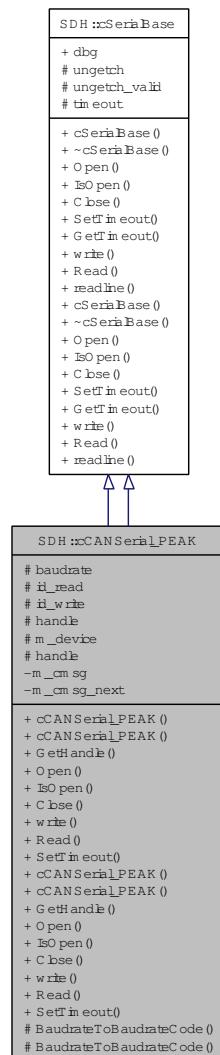
The documentation for this class was generated from the following files:

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

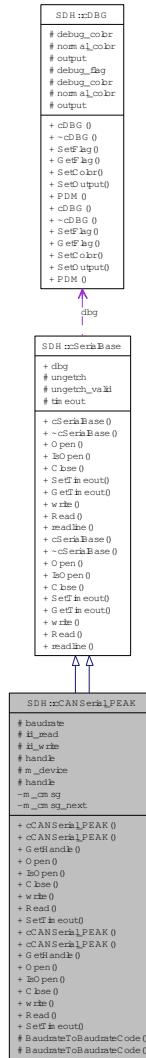
## 10.3 SDH::cCANSerial\_PEAK Class Reference

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

Inheritance diagram for SDH::cCANSerial\_PEAK:



Collaboration diagram for SDH::cCANSerial\_PEEK:



### 10.3.1 Detailed Description

Low-level communication class to access a CAN port.

#### Public Member Functions

- **cCANSerial\_PEEK** (unsigned long \_baudrate, double \_timeout, int32\_t \_id\_read, int32\_t \_id\_write, const char \*device="/dev/pcanusb0") throw (cCANSerial\_PEEKException\*)
- **cCANSerial\_PEEK** (HANDLE \_handle, double \_timeout, int32\_t \_id\_read, int32\_t \_id\_write) throw (cCANSerial\_PEEKException\*)
- HANDLE **GetHandle** ()
- void **Open** (void) throw (cCANSerial\_PEEKException\*)
- bool **IsOpen** (void) throw ()

*Return true if interface to CAN PEAK is open.*

- void **Close** (void) throw (cCANSerial\_PEAKException\*)
 

*Close the previously opened CAN PEAK interface port.*
- int **write** (char const \*ptr, int len=0) throw (cCANSerial\_PEAKException\*)
 

*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\_PEAKException\*)
 

*(cCANSerial\_PEAKException\*)*
- void **SetTimeout** (double \_timeout) throw (cSerialBaseException\*)
 

*set the timeout for next `readline()` calls (negative value means: no timeout, wait for ever)*
- **cCANSerial\_PEAK** (unsigned long \_baudrate, double \_timeout, Int32 \_id\_read, Int32 \_id\_write, const char \*device="/dev/pcanusb0") throw (cCANSerial\_PEAKException\*)
 

*(cCANSerial\_PEAKException\*)*
- **cCANSerial\_PEAK (PCAN\_HANDLE \_handle, double \_timeout, Int32 \_id\_read, Int32 \_id\_write)** throw (cCANSerial\_PEAKException\*)
 

*(cCANSerial\_PEAKException\*)*
- **PCAN\_HANDLE GetHandle ()**
- void **Open** (void) throw (cCANSerial\_PEAKException\*)
 

*(cCANSerial\_PEAKException\*)*
- bool **IsOpen** (void) throw ()
 

*Return true if interface to CAN PEAK is open.*
- void **Close** (void) throw (cCANSerial\_PEAKException\*)
 

*Close the previously opened CAN PEAK interface port.*
- int **write** (char const \*ptr, int len=0) throw (cCANSerial\_PEAKException\*)
 

*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\_PEAKException\*)
 

*(cCANSerial\_PEAKException\*)*
- void **SetTimeout** (double \_timeout) throw (cSerialBaseException\*)
 

*set the timeout for next `readline()` calls (negative value means: no timeout, wait for ever)*

## Protected Member Functions

- WORD **BaudrateToBaudrateCode** (unsigned long **baudrate**) throw (cCANSerial\_PEAKException\*)
 

*Translate a baudrate given as unsigned long into a baudrate code for struct termios.*
- WORD **BaudrateToBaudrateCode** (unsigned long **baudrate**) throw (cCANSerial\_PEAKException\*)
 

*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*

- DWORD `id_read`  
*the CAN ID used for reading*
- DWORD `id_write`  
*the CAN ID used for writing*
- HANDLE `handle`  
*the handle to the driver*
- char `m_device` [64]
- PCAN\_HANDLE `handle`  
*the handle to the driver*

### 10.3.2 Constructor & Destructor Documentation

**10.3.2.1 cCANSerial\_Peak::cCANSerial\_Peak (unsigned long `_baudrate`, double `_timeout`, int32\_t `_id_read`, int32\_t `_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:**

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

**10.3.2.2 cCANSerial\_Peak::cCANSerial\_Peak (HANDLE `_handle`, double `_timeout`, int32\_t `_id_read`, int32\_t `_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:**

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

---

**10.3.2.3 SDH::cCANSerial\_Peak::cCANSerial\_Peak (*unsigned long \_baudrate, double \_timeout, Int32 \_id\_read, Int32 \_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:**

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

**10.3.2.4 cCANSerial\_Peak::cCANSerial\_Peak (PCAN\_HANDLE *\_handle, double \_timeout, Int32 \_id\_read, Int32 \_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:**

*\_handle* - the PEAK CAN handle to reuse (Works on Linux only!)  
*\_timeout* - the timeout in seconds (0 for no timeout = wait for ever)  
*\_id\_read* - the CAN ID to use for reading (The [SDH](#) sends data on this ID)  
*\_id\_write* - the CAN ID to use for writing (The [SDH](#) receives data on this ID)

### 10.3.3 Member Function Documentation

**10.3.3.1 WORD cCANSerial\_Peak::BaudrateToBaudrateCode (*unsigned long baudrate*) throw (cCANSerial\_PeakException\*) [protected]**

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

**10.3.3.2 HANDLE SDH::cCANSerial\_Peak::GetHandle () [inline]**

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

---

**10.3.3.3 void cCANSerial\_PEAKE::Open (void) throw (cCANSerial\_PEAKEException\*) [virtual]**

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

Implements [SDH::cSerialBase](#).

**10.3.3.4 bool cCANSerial\_PEAKE::IsOpen (void) throw () [virtual]**

Return true if interface to CAN PEAK is open.

Implements [SDH::cSerialBase](#).

**10.3.3.5 void cCANSerial\_PEAKE::Close (void) throw (cCANSerial\_PEAKEException\*) [virtual]**

Close the previously opened CAN PEAK interface port.

Implements [SDH::cSerialBase](#).

**10.3.3.6 int cCANSerial\_PEAKE::write (char const \* ptr, int len = 0) throw (cCANSerial\_PEAKEException\*) [virtual]**

Write data to a previously opened port.

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

**Parameters:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

**10.3.3.7 ssize\_t cCANSerial\_PEAKE::Read (void \* data, ssize\_t size, long timeout\_us, bool return\_on\_less\_data) throw (cCANSerial\_PEAKEException\*) [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](#).

**10.3.3.8 void cCANSerial\_PEAKE::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](#).

**10.3.3.9 WORD SDH::cCANSerial\_PEAK::BaudrateToBaudrateCode (unsigned long *baudrate*)  
throw (cCANSerial\_PeakException\*) [protected]**

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

**10.3.3.10 PCAN\_HANDLE SDH::cCANSerial\_PEAK::GetHandle () [inline]**

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

**10.3.3.11 void SDH::cCANSerial\_PEAK::Open (void) throw (cCANSerial\_PeakException\*)  
[virtual]**

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

Implements [SDH::cSerialBase](#).

**10.3.3.12 bool SDH::cCANSerial\_PEAK::IsOpen (void) throw () [virtual]**

Return true if interface to CAN PEAK is open.

Implements [SDH::cSerialBase](#).

**10.3.3.13 void SDH::cCANSerial\_PEAK::Close (void) throw (cCANSerial\_PeakException\*)  
[virtual]**

Close the previously opened CAN PEAK interface port.

Implements [SDH::cSerialBase](#).

**10.3.3.14 int SDH::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:**

*ptr* - pointer the byte array to send in memory  
*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

---

**10.3.3.15 ssize\_t SDH::cCANSerial\_PEAKE::Read (void \* *data*, ssize\_t *size*, long *timeout\_us*, bool *return\_on\_less\_data*) throw (cCANSerial\_PEAKEException\*) [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](#).

**10.3.3.16 void SDH::cCANSerial\_PEAKE::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](#).

## 10.3.4 Member Data Documentation

**10.3.4.1 unsigned long SDH::cCANSerial\_PEAKE::baudrate [protected]**

the baudrate to use in bit/s

**10.3.4.2 DWORD SDH::cCANSerial\_PEAKE::id\_read [protected]**

the CAN ID used for reading

**10.3.4.3 DWORD SDH::cCANSerial\_PEAKE::id\_write [protected]**

the CAN ID used for writing

**10.3.4.4 HANDLE SDH::cCANSerial\_PEAKE::handle [protected]**

the handle to the driver

**10.3.4.5 char SDH::cCANSerial\_PEAKE::m\_device [protected]**
**10.3.4.6 PCAN\_HANDLE SDH::cCANSerial\_PEAKE::handle [protected]**

the handle to the driver

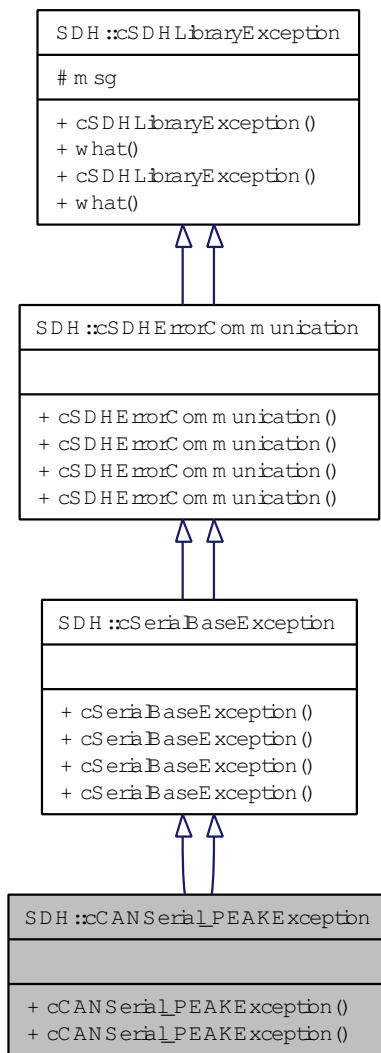
The documentation for this class was generated from the following files:

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

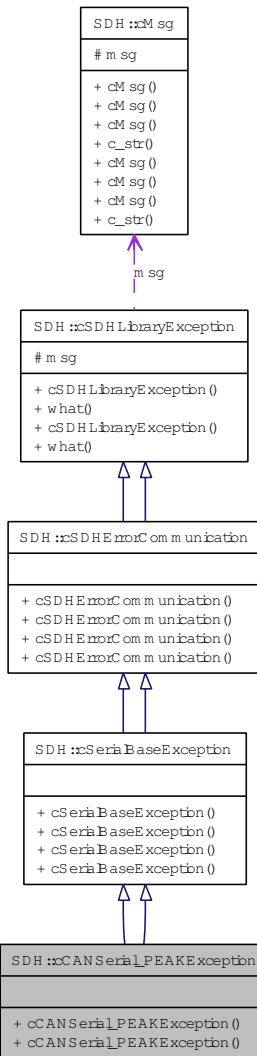
## 10.4 SDH::cCANSerial\_PEAKEexception Class Reference

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

Inheritance diagram for SDH::cCANSerial\_PEAKEexception:



Collaboration diagram for SDH::cCANSerial\_PEAKEexception:



#### 10.4.1 Detailed Description

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

#### Public Member Functions

- [cCANSerial\\_PEAKEexception \(cMsg const &\\_msg\)](#)
- [cCANSerial\\_PEAKEexception \(cMsg const &\\_msg\)](#)

## 10.4.2 Constructor & Destructor Documentation

**10.4.2.1 SDH::cCANSerial\_PEAKException::cCANSerial\_PEAKException (cMsg const & *\_msg*)**  
[inline]

**10.4.2.2 SDH::cCANSerial\_PEAKException::cCANSerial\_PEAKException (cMsg const & *\_msg*)**  
[inline]

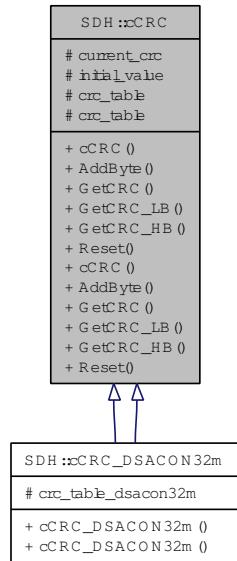
The documentation for this class was generated from the following files:

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

## 10.5 SDH::cCRC Class Reference

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

Inheritance diagram for SDH::cCRC:



### 10.5.1 Detailed Description

Generic class to calculate a CRC using a given, precalculated table.

Use derived classes like [cCRC\\_DSAON32m](#) with a specifically set CRC table.

### 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 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;*

- **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 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*
- **tCRCValue const \* *crc\_table***  
*table with precalculated CRC values*

### 10.5.2 Constructor & Destructor Documentation

#### 10.5.2.1 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.

#### 10.5.2.2 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.

### 10.5.3 Member Function Documentation

#### 10.5.3.1 tCRCValue SDH::cCRC::AddByte (unsigned char byte) [inline]

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

#### 10.5.3.2 tCRCValue SDH::cCRC::GetCRC () [inline]

return the current CRC value

#### 10.5.3.3 UInt8 SDH::cCRC::GetCRC\_LB () [inline]

return the low byte of the current CRC value

#### 10.5.3.4 UInt8 SDH::cCRC::GetCRC\_HB () [inline]

return the high byte of the current CRC value

#### 10.5.3.5 tCRCValue SDH::cCRC::Reset () [inline]

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

#### 10.5.3.6 tCRCValue SDH::cCRC::AddByte (unsigned char byte) [inline]

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

#### 10.5.3.7 tCRCValue SDH::cCRC::GetCRC () [inline]

return the current CRC value

#### 10.5.3.8 UInt8 SDH::cCRC::GetCRC\_LB () [inline]

return the low byte of the current CRC value

#### 10.5.3.9 UInt8 SDH::cCRC::GetCRC\_HB () [inline]

return the high byte of the current CRC value

#### 10.5.3.10 tCRCValue SDH::cCRC::Reset () [inline]

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

### 10.5.4 Member Data Documentation

#### 10.5.4.1 tCRCValue SDH::cCRC::current\_crc [protected]

current value of the CRC checksum

**10.5.4.2 tCRCValue SDH::cCRC::initial\_value [protected]**

initial value of the CRC checksum

**10.5.4.3 tCRCValue const\* SDH::cCRC::crc\_table [protected]**

table with precalculated CRC values

**10.5.4.4 tCRCValue const\* SDH::cCRC::crc\_table [protected]**

table with precalculated CRC values

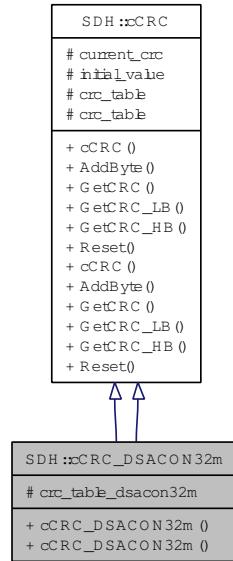
The documentation for this class was generated from the following files:

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

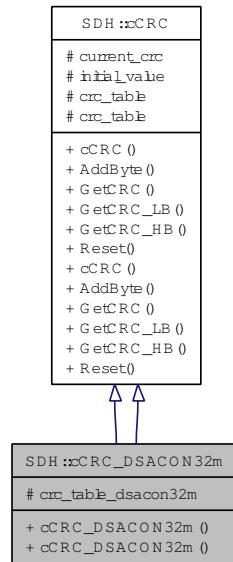
## 10.6 SDH::cCRC\_DSACON32m Class Reference

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

Inheritance diagram for SDH::cCRC\_DSACON32m:



Collaboration diagram for SDH::cCRC\_DSACON32m:



### 10.6.1 Detailed Description

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

## Public Member Functions

- **cCRC\_DSACON32m (void)**

*constructor to create a **cCRC** object suitable for checksumming the communication with a DSACON32m tactile sensor controller*

- **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*

### 10.6.2 Constructor & Destructor Documentation

#### 10.6.2.1 SDH::cCRC\_DSACON32m::cCRC\_DSACON32m (void) [inline]

constructor to create a **cCRC** object suitable for checksumming the communication with a DSACON32m tactile sensor controller

#### 10.6.2.2 SDH::cCRC\_DSACON32m::cCRC\_DSACON32m (void) [inline]

constructor to create a **cCRC** object suitable for checksumming the communication with a DSACON32m tactile sensor controller

### 10.6.3 Member Data Documentation

#### 10.6.3.1 static tCRCValue const SDH::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, 0xa50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbdc, 0xfbff, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    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, 0xb1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xaleb, 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:

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

## 10.7 SDH::cDBG Class Reference

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

### 10.7.1 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 representation: " << 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";
```

### 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))
- [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`
- char const \* `debug_color`
- char const \* `normal_color`
- std::ostream \* `output`

### 10.7.2 Constructor & Destructor Documentation

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

constructor: construct a `cDBG` object

#### Parameters:

*flag* - the initial state of the flag, if true then messages sent to the object are printed. Default is false.  
Can be changed with [SetFlag\(\)](#)

*color* - the name of the color to use, default is "red". Can be changed with [SetColor\(\)](#)

*fd* - the ostream to use for output, default is stderr. Can be changed with [SetOutput\(\)](#)

**10.7.2.2 SDH::cDBG::~cDBG () [inline]**

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

constructor: construct a `cDBG` object

#### Parameters:

*flag* - the initial state of the flag, if true then messages sent to the object are printed. Default is false.  
Can be changed with [SetFlag\(\)](#)

*color* - the name of the color to use, default is "red". Can be changed with [SetColor\(\)](#)

*fd* - the ostream to use for output, default is stderr. Can be changed with [SetOutput\(\)](#)

**10.7.2.4 SDH::cDBG::~cDBG () [inline]**

### 10.7.3 Member Function Documentation

**10.7.3.1 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.

**10.7.3.2 bool SDH::cDBG::GetFlag (void) const [inline]**

Get debug\_flag of this **cDBG** object.

**10.7.3.3 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

**10.7.3.4 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

**10.7.3.5 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.

**10.7.3.6 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.

**10.7.3.7 bool SDH::cDBG::GetFlag (void) const [inline]**

Get debug\_flag of this **cDBG** object.

**10.7.3.8 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

**10.7.3.9 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

**10.7.3.10 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.

## 10.7.4 Member Data Documentation

**10.7.4.1** `char const* SDH::cDBG::debug_color` [protected]

**10.7.4.2** `char const* SDH::cDBG::normal_color` [protected]

**10.7.4.3** `std::ostream* SDH::cDBG::output` [protected]

**10.7.4.4** `bool SDH::cDBG::debug_flag` [protected]

**10.7.4.5** `char const* SDH::cDBG::debug_color` [protected]

**10.7.4.6** `char const* SDH::cDBG::normal_color` [protected]

**10.7.4.7** `std::ostream* SDH::cDBG::output` [protected]

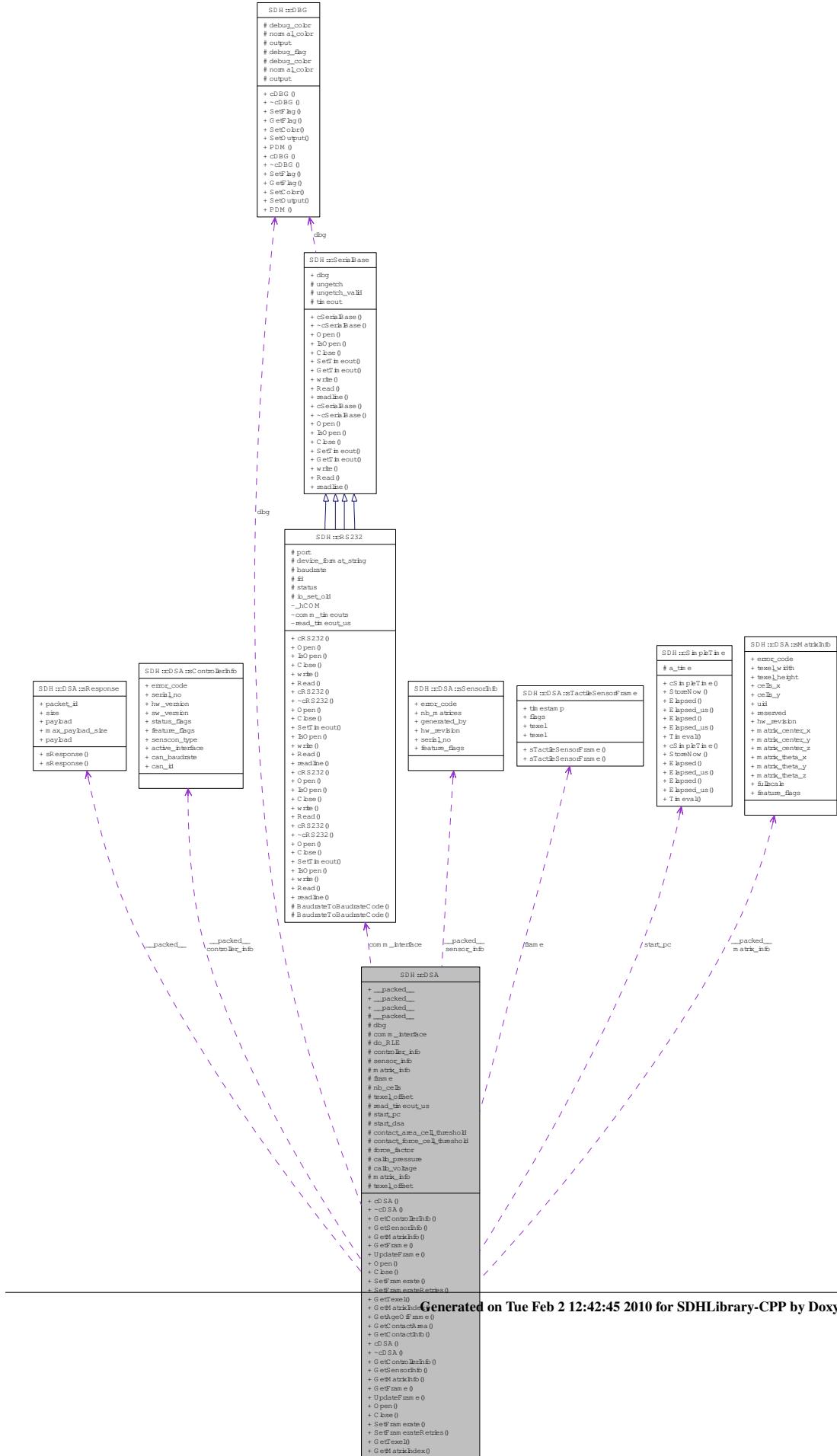
The documentation for this class was generated from the following files:

- `cpp.desire.final/sdh/dbg.h`
- `sdh/dbg.h`

## **10.8 SDH::cDSA Class Reference**

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

## Collaboration diagram for SDH::cDSA:



### 10.8.1 Detailed Description

[SDH::cDSA](#) is the end user interface class to access the DSACON32m, the tactile sensor controller of the SDH.

Class to communicate with the tactile sensor controller DSACON32m of the SDH

#### Bug

[cDSAEexception](#): Checksum Error on Windows console While communicating with the tactile sensor controller a "cDSAEexception: 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.

Class to communicate with the tactile sensor controller DSACON32m of the SDH

#### Bug

[cDSAEexception](#): Checksum Error on Windows console While communicating with the tactile sensor controller a "cDSAEexception: 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.

## Public Types

- `typedef UInt16 tTexel`  
*data type for a single 'texel' (tactile sensor element)*
- `typedef UInt16 tTexel`  
*data type for a single 'texel' (tactile sensor element)*

## Public Member Functions

- `cDSA` (int debug\_level=0, int port=1, char const \*device\_format\_string="/dev/ttyS%d")  
`~cDSA ()`  
*Destructor: clean up and delete dynamically allocated memory.*
- `sControllerInfo const & GetControllerInfo () const`  
*Return a reference to the `sControllerInfo` read from the remote DSACON32m controller.*
- `sSensorInfo const & GetSensorInfo () 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)
 

*(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)
 

*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)
- **void SetFramerateRetries** (**UInt16** framerate, bool **do\_RLE**=true, bool **do\_data\_acquisition**=true, unsigned int retries=0, bool ignore\_exceptions=false) throw (**cDSAException**\*)
- **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)
- **cDSA** (int debug\_level=0, int port=1, char const \*device\_format\_string="/dev/ttyS%d")
- **~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)
 

*(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)
 

*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)
- **void SetFramerateRetries** (**UInt16** framerate, bool **do\_RLE**=true, bool **do\_data\_acquisition**=true, unsigned int retries=0, bool ignore\_exceptions=false) throw (**cDSAException**\*)
- **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`)

## Public Attributes

- struct `SDH::cDSA::sControllerInfo __packed__`
- struct `SDH::cDSA::sSensorInfo __packed__`
- struct `SDH::cDSA::sMatrixInfo __packed__`

## Protected Member Functions

- void `WriteCommandWithPayload` (`UInt8` `command`, `UInt8` \*`payload`, `UInt16` `payload_len`)
- void `WriteCommand` (`UInt8` `command`)
- void `ReadResponse` (`sResponse` \*`response`)
- void `ReadControllerInfo` (`sControllerInfo` \*`_controller_info`)
- void `ReadSensorInfo` (`sSensorInfo` \*`_sensor_info`)
- void `ReadMatrixInfo` (`sMatrixInfo` \*`_matrix_info`)
- void `ReadFrame` (`sTactileSensorFrame` \*`frame_p`)
- void `QueryControllerInfo` (`sControllerInfo` \*`_controller_info`)
- void `QuerySensorInfo` (`sSensorInfo` \*`_sensor_info`)
- void `QueryMatrixInfo` (`sMatrixInfo` \*`_matrix_info`, int `matrix_no`)
- void `QueryMatrixInfos` (void)
- void `ParseFrame` (`sResponse` \*`response`, `sTactileSensorFrame` \*`frame_p`)
- void `WriteCommandWithPayload` (`UInt8` `command`, `UInt8` \*`payload`, `UInt16` `payload_len`)
- void `WriteCommand` (`UInt8` `command`)
- void `ReadResponse` (`sResponse` \*`response`)
- void `ReadControllerInfo` (`sControllerInfo` \*`_controller_info`)
- void `ReadSensorInfo` (`sSensorInfo` \*`_sensor_info`)
- void `ReadMatrixInfo` (`sMatrixInfo` \*`_matrix_info`)
- void `ReadFrame` (`sTactileSensorFrame` \*`frame_p`)
- void `QueryControllerInfo` (`sControllerInfo` \*`_controller_info`)
- void `QuerySensorInfo` (`sSensorInfo` \*`_sensor_info`)
- void `QueryMatrixInfo` (`sMatrixInfo` \*`_matrix_info`, int `matrix_no`)
- void `QueryMatrixInfos` (void)
- void `ParseFrame` (`sResponse` \*`response`, `sTactileSensorFrame` \*`frame_p`)

## Protected Attributes

- struct `SDH::cDSA::sResponse __packed__`
- `cDBG dbg`

*A stream object to print coloured debug messages.*

- `cRS232 comm_interface`

*the serial RS232 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*
- **sMatrixInfo \* matrix\_info**  
*the matrix infos read from the remote DSACON32m controller*
- **int \* texel\_offset**  
*an array with the offsets of the first texel of all matrices into the whole frame*

## Friends

- std::ostream & **operator<<** (std::ostream &stream, **cDSA::sResponse** const &response)
- std::ostream & **operator<<** (std::ostream &stream, **cDSA::sResponse** const &response)

## 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 [sSensorInfo](#)  
*A data structure describing the sensor info about the remote DSACON32m controller.*
- struct [sTactileSensorFrame](#)

### 10.8.2 Member Typedef Documentation

#### 10.8.2.1 [typedef UInt16 SDH::cDSA::tTexel](#)

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

#### 10.8.2.2 [typedef UInt16 SDH::cDSA::tTexel](#)

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

### 10.8.3 Constructor & Destructor Documentation

#### 10.8.3.1 [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 DSACON32m controller within the SDH.

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:

*debug\_level* - the level of debug messages to be printed:

- 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

*port* - the RS232 port to use for communication. (port 0 = ttyS0 = COM1, port 1 = ttyS1 = COM2, ...)

*device\_format\_string* - 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.

#### 10.8.3.2 cDSA::~cDSA ()

Destructor: clean up and delete dynamically allocated memory.

#### 10.8.3.3 SDH::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 DSACON32m controller within the SDH.

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:

*debug\_level* - the level of debug messages to be printed:

- 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

*port* - the RS232 port to use for communication. (port 0 = ttyS0 = COM1, port 1 = ttyS1 = COM2, ...)

*device\_format\_string* - 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.

#### 10.8.3.4 SDH::cDSA::~cDSA ()

Destructor: clean up and delete dynamically allocated memory.

### 10.8.4 Member Function Documentation

#### 10.8.4.1 void cDSA::WriteCommandWithPayload (UInt8 *command*, UInt8 \* *payload*, UInt16 *payload\_len*) [protected]

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

#### 10.8.4.3 void cDSA::ReadResponse (sResponse \* *response*) [protected]

Read any response from the remote DSACON32m

- 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

**10.8.4.4 void cDSA::ReadControllerInfo (sControllerInfo \* *\_controller\_info*) [protected]**

Read and parse a controller info response from the remote DSA

**10.8.4.5 void cDSA::ReadSensorInfo (sSensorInfo \* *\_sensor\_info*) [protected]**

Read and parse a sensor info response from the remote DSA

**10.8.4.6 void cDSA::ReadMatrixInfo (sMatrixInfo \* *\_matrix\_info*) [protected]**

Read and parse a matrix info response from the remote DSA

**10.8.4.7 void cDSA::ReadFrame (sTactileSensorFrame \* *frame\_p*) [protected]**

read and parse a full frame response from remote DSA

**10.8.4.8 void cDSA::QueryControllerInfo (sControllerInfo \* *\_controller\_info*) [protected]**

Send command to request controller info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.9 void cDSA::QuerySensorInfo (sSensorInfo \* *\_sensor\_info*) [protected]**

Send command to request sensor info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.10 void cDSA::QueryMatrixInfo (sMatrixInfo \* *\_matrix\_info*, int *matrix\_no*) [protected]**

Send command to request matrix info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.11 void cDSA::QueryMatrixInfos (void) [protected]**

Query all matrix infos

---

**10.8.4.12 void cDSA::ParseFrame (sResponse \* *response*, sTactileSensorFrame \* *frame\_p*) [protected]**

Parse a full frame response from remote DSA

**10.8.4.13 sControllerInfo const& SDH::cDSA::GetControllerInfo (void) const [inline]**

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

**10.8.4.14 sSensorInfo const& SDH::cDSA::GetSensorInfo (void) const [inline]**

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

**10.8.4.15 sMatrixInfo const& SDH::cDSA::GetMatrixInfo (int *m*) const [inline]**

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

**10.8.4.16 sTactileSensorFrame const& SDH::cDSA::GetFrame () const [inline]**

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

**10.8.4.17 sTactileSensorFrame const& SDH::cDSA::UpdateFrame () [inline]**

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.

**10.8.4.18 void cDSA::Open (void)**

(Re-)open connection to DSACON32m controller, this is called by the constructor automatically, but is still usefull to call after a call to [Close\(\)](#)

**10.8.4.19 void cDSA::Close (void)**

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

**10.8.4.20 void cDSA::SetFramerate (UInt16 *framerate*, bool *do\_RLE* = true, bool *do\_data\_acquisition* = true)**

Set the *framerate* for querying full frames.

**Parameters:**

*framerate* - 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)).

*do\_RLE* - flag, if true then use RLE (run length encoding) for sending frames

*do\_data\_acquisition* - flag, enable or disable data acquisition. Must be true if you want to get new frames

---

**10.8.4.21 void cDSA::SetFramerateRetries (UInt16 *framerate*, bool *do\_RLE* = true, bool *do\_data\_acquisition* = true, unsigned int *retries* = 0, bool *ignore\_exceptions* = false)  
throw (cDSAException\*)**

Set the *framerate* for querying full frames.

**Parameters:**

*framerate* - 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)).

*do\_RLE* - flag, if true then use RLE (run length encoding) for sending frames

*do\_data\_acquisition* - flag, enable or disable data acquisition. Must be true if you want to get new frames

*retries* - number of times the sending will be retried in case of an error (like timeout while waiting for response)

*ignore\_exceptions* - flag, if true then exceptions are ignored in case of error. After *retries* tries the function just returns even in case of an error

**10.8.4.22 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

**10.8.4.23 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]

**10.8.4.24 unsigned long SDH::cDSA::GetAgeOfFrame (sTactileSensorFrame \**frame\_p*) [inline]**

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

**10.8.4.25 double cDSA::GetContactArea (int *m*)**

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

**10.8.4.26 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*

---

**10.8.4.27 void SDH::cDSA::WriteCommandWithPayload (UInt8 *command*, UInt8 \* *payload*, UInt16 *payload\_len*) [protected]**

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

**10.8.4.29 void SDH::cDSA::ReadResponse (sResponse \* *response*) [protected]**

Read any response from the remote DSACON32m

- 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

**10.8.4.30 void SDH::cDSA::ReadControllerInfo (sControllerInfo \* *\_controller\_info*) [protected]**

Read and parse a controller info response from the remote DSA

**10.8.4.31 void SDH::cDSA::ReadSensorInfo (sSensorInfo \* *\_sensor\_info*) [protected]**

Read and parse a sensor info response from the remote DSA

**10.8.4.32 void SDH::cDSA::ReadMatrixInfo (sMatrixInfo \* *\_matrix\_info*) [protected]**

Read and parse a matrix info response from the remote DSA

**10.8.4.33 void SDH::cDSA::ReadFrame (sTactileSensorFrame \* *frame\_p*) [protected]**

read and parse a full frame response from remote DSA

**10.8.4.34 void SDH::cDSA::QueryControllerInfo (sControllerInfo \* *\_controller\_info*) [protected]**

Send command to request controller info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.35 void SDH::cDSA::QuerySensorInfo (sSensorInfo \* *\_sensor\_info*) [protected]**

Send command to request sensor info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.36 void SDH::cDSA::QueryMatrixInfo (sMatrixInfo \* *\_matrix\_info*, int *matrix\_no*) [protected]**

Send command to request matrix info from remote DSACON32m. Read and parse the response from the remote DSACON32m.

**10.8.4.37 void SDH::cDSA::QueryMatrixInfos (void) [protected]**

Query all matrix infos

**10.8.4.38 void SDH::cDSA::ParseFrame (sResponse \* *response*, sTactileSensorFrame \* *frame\_p*) [protected]**

Parse a full frame response from remote DSA

**10.8.4.39 sControllerInfo const& SDH::cDSA::GetControllerInfo (void) const [inline]**

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

**10.8.4.40 sSensorInfo const& SDH::cDSA::GetSensorInfo (void) const [inline]**

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

**10.8.4.41 sMatrixInfo const& SDH::cDSA::GetMatrixInfo (int *m*) const [inline]**

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

**10.8.4.42 sTactileSensorFrame const& SDH::cDSA::GetFrame () const [inline]**

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

**10.8.4.43 sTactileSensorFrame const& SDH::cDSA::UpdateFrame () [inline]**

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.

**10.8.4.44 void SDH::cDSA::Open (void)**

(Re-)open connection to DSACON32m controller, this is called by the constructor automatically, but is still usefull to call after a call to [Close\(\)](#)

**10.8.4.45 void SDH::cDSA::Close (void)**

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

---

**10.8.4.46 void SDH::cDSA::SetFramerate (UInt16 *framerate*, bool *do\_RLE* = true, bool *do\_data\_acquisition* = true)**

Set the *framerate* for querying full frames.

**Parameters:**

*framerate* - 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)).

*do\_RLE* - flag, if true then use RLE (run length encoding) for sending frames

*do\_data\_acquisition* - flag, enable or disable data acquisition. Must be true if you want to get new frames

---

**10.8.4.47 void SDH::cDSA::SetFramerateRetries (UInt16 *framerate*, bool *do\_RLE* = true, bool *do\_data\_acquisition* = true, unsigned int *retries* = 0, bool *ignore\_exceptions* = false) throw (cDSAException\*)**

Set the *framerate* for querying full frames.

**Parameters:**

*framerate* - 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)).

*do\_RLE* - flag, if true then use RLE (run length encoding) for sending frames

*do\_data\_acquisition* - flag, enable or disable data acquisition. Must be true if you want to get new frames

*retries* - number of times the sending will be retried in case of an error (like timeout while waiting for response)

*ignore\_exceptions* - flag, if true then exceptions are ignored in case of error. After *retries* tries the function just returns even in case of an error

---

**10.8.4.48 tTexel SDH::cDSA::GetTexel (int *m*, int *x*, int *y*) const**

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

**10.8.4.49 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]

**10.8.4.50 unsigned long SDH::cDSA::GetAgeOfFrame (sTactileSensorFrame \**frame\_p*) [inline]**

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

**10.8.4.51 double SDH::cDSA::GetContactArea (int *m*)**

**10.8.4.52 sContactInfo SDH::cDSA::GetContactInfo (int *m*)**

## 10.8.5 Friends And Related Function Documentation

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

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

## 10.8.6 Member Data Documentation

**10.8.6.1 struct SDH::cDSA::sResponse SDH::cDSA::\_\_packed\_\_**

**10.8.6.2 struct SDH::cDSA::sSensorInfo SDH::cDSA::\_\_packed\_\_**

**10.8.6.3 struct SDH::cDSA::sMatrixInfo SDH::cDSA::\_\_packed\_\_**

**10.8.6.4 struct SDH::cDSA::sResponse SDH::cDSA::\_\_packed\_\_ [protected]**

**10.8.6.5 cDBG SDH::cDSA::dbg [protected]**

A stream object to print coloured debug messages.

**10.8.6.6 cRS232 SDH::cDSA::comm\_interface [protected]**

the serial RS232 communication interface to use

**10.8.6.7 bool SDH::cDSA::do\_RLE [protected]**

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

**10.8.6.8 sControllerInfo SDH::cDSA::controller\_info [protected]**

the controller info read from the remote DSACON32m controller

**10.8.6.9 sSensorInfo SDH::cDSA::sensor\_info [protected]**

the sensor info read from the remote DSACON32m controller

**10.8.6.10 sMatrixInfo\* SDH::cDSA::matrix\_info [protected]**

the matrix infos read from the remote DSACON32m controller

**10.8.6.11 sTactileSensorFrame SDH::cDSA::frame [protected]**

the latest frame received from the remote DSACON32m controller

**10.8.6.12 int SDH::cDSA::nb\_cells [protected]**

The total number of sensor cells.

**10.8.6.13 int\* SDH::cDSA::texel\_offset [protected]**

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

**10.8.6.14 long SDH::cDSA::read\_timeout\_us [protected]**

default timeout used for reading in us

**10.8.6.15 cSimpleTime SDH::cDSA::start\_pc [protected]****10.8.6.16 UInt32 SDH::cDSA::start\_dsa [protected]****10.8.6.17 tTexel SDH::cDSA::contact\_area\_cell\_threshold [protected]**

threshold of texel cell value for detecting contacts with GetContactArea

**10.8.6.18 tTexel SDH::cDSA::contact\_force\_cell\_threshold [protected]**

threshold of texel cell value for detecting forces with GetContactForce

**10.8.6.19 double SDH::cDSA::force\_factor [protected]**

additional calibration factor for forces in GetContactForce

**10.8.6.20 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):

**10.8.6.21 double SDH::cDSA::calib\_voltage [protected]**

see calib\_pressure

**10.8.6.22 sMatrixInfo\* SDH::cDSA::matrix\_info [protected]**

the matrix infos read from the remote DSACON32m controller

**10.8.6.23 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:

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

- `sdh/dsa.h`
- `cpp.desire.final/sdh/dsa.cpp`
- `sdh/dsa.cpp`

## 10.9 SDH::cDSA::sContactInfo Struct Reference

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

### 10.9.1 Detailed Description

Structure to hold info about the contact of one sensor patch.

#### 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*

### 10.9.2 Member Data Documentation

#### 10.9.2.1 double SDH::cDSA::sContactInfo::force

accumulated force in N

#### 10.9.2.2 double SDH::cDSA::sContactInfo::area

area of contact in mm\*mm.

#### 10.9.2.3 double SDH::cDSA::sContactInfo::cog\_x

center of gravity of contact in x direction of sensor patch in mm

#### 10.9.2.4 double SDH::cDSA::sContactInfo::cog\_y

center of gravity of contact in y direction of sensor patch in mm

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

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

## 10.10 SDH::cDSA::sControllerInfo Struct Reference

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

### 10.10.1 Detailed Description

A data structure describing the controller info about the remote DSACON32m controller.

#### 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`

### 10.10.2 Member Data Documentation

**10.10.2.1 UInt16 SDH::cDSA::sControllerInfo::error\_code**

**10.10.2.2 UInt32 SDH::cDSA::sControllerInfo::serial\_no**

**10.10.2.3 UInt8 SDH::cDSA::sControllerInfo::hw\_version**

**10.10.2.4 UInt16 SDH::cDSA::sControllerInfo::sw\_version**

**10.10.2.5 UInt8 SDH::cDSA::sControllerInfo::status\_flags**

**10.10.2.6 UInt8 SDH::cDSA::sControllerInfo::feature\_flags**

**10.10.2.7 UInt8 SDH::cDSA::sControllerInfo::senscon\_type**

**10.10.2.8 UInt8 SDH::cDSA::sControllerInfo::active\_interface**

**10.10.2.9 UInt32 SDH::cDSA::sControllerInfo::can\_baudrate**

**10.10.2.10 UInt16 SDH::cDSA::sControllerInfo::can\_id**

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

- `cpp.desire.final/sdh/dsa.h`
- `sdh/dsa.h`

## 10.11 SDH::cDSA::sMatrixInfo Struct Reference

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

### 10.11.1 Detailed Description

A data structure describing a single sensor matrix connected to the remote DSACON32m controller.

### 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](#)

## 10.11.2 Member Data Documentation

**10.11.2.1 UInt16 SDH::cDSA::sMatrixInfo::error\_code**

**10.11.2.2 float SDH::cDSA::sMatrixInfo::texel\_width**

**10.11.2.3 float SDH::cDSA::sMatrixInfo::texel\_height**

**10.11.2.4 UInt16 SDH::cDSA::sMatrixInfo::cells\_x**

**10.11.2.5 UInt16 SDH::cDSA::sMatrixInfo::cells\_y**

**10.11.2.6 UInt8 SDH::cDSA::sMatrixInfo::uid**

**10.11.2.7 UInt8 SDH::cDSA::sMatrixInfo::reserved**

**10.11.2.8 UInt8 SDH::cDSA::sMatrixInfo::hw\_revision**

**10.11.2.9 float SDH::cDSA::sMatrixInfo::matrix\_center\_x**

**10.11.2.10 float SDH::cDSA::sMatrixInfo::matrix\_center\_y**

**10.11.2.11 float SDH::cDSA::sMatrixInfo::matrix\_center\_z**

**10.11.2.12 float SDH::cDSA::sMatrixInfo::matrix\_theta\_x**

**10.11.2.13 float SDH::cDSA::sMatrixInfo::matrix\_theta\_y**

**10.11.2.14 float SDH::cDSA::sMatrixInfo::matrix\_theta\_z**

**10.11.2.15 float SDH::cDSA::sMatrixInfo::fullscale**

**10.11.2.16 UInt8 SDH::cDSA::sMatrixInfo::feature\_flags**

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

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

## 10.12 SDH::cDSA::sResponse Struct Reference

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

### 10.12.1 Detailed Description

data structure for storing responses from the remote DSACON32m controller

### Public Member Functions

- [sResponse \(UInt8 \\*\\_payload, int \\_max\\_payload\\_size\)](#)

*constructor to init pointer and max size*

- [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](#)
- [UInt8 \\* payload](#)

### 10.12.2 Constructor & Destructor Documentation

#### 10.12.2.1 SDH::cDSA::sResponse::sResponse (UInt8 \* *\_payload*, int *\_max\_payload\_size*) [inline]

constructor to init pointer and max size

#### 10.12.2.2 SDH::cDSA::sResponse::sResponse (UInt8 \* *\_payload*, int *\_max\_payload\_size*) [inline]

constructor to init pointer and max size

### 10.12.3 Member Data Documentation

**10.12.3.1 UInt8 SDH::cDSA::sResponse::packet\_id**

**10.12.3.2 UInt16 SDH::cDSA::sResponse::size**

**10.12.3.3 UInt8\* SDH::cDSA::sResponse::payload**

**10.12.3.4 Int32 SDH::cDSA::sResponse::max\_payload\_size**

**10.12.3.5 UInt8\* SDH::cDSA::sResponse::payload**

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

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

## 10.13 SDH::cDSA::sSensorInfo Struct Reference

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

### 10.13.1 Detailed Description

A data structure describing the sensor info about the remote DSACON32m controller.

#### Public Attributes

- [UInt16 error\\_code](#)
- [UInt16 nb\\_matrices](#)
- [UInt16 generated\\_by](#)
- [UInt8 hw\\_revision](#)
- [UInt32 serial\\_no](#)
- [UInt8 feature\\_flags](#)

### 10.13.2 Member Data Documentation

**10.13.2.1 UInt16 SDH::cDSA::sSensorInfo::error\_code**

**10.13.2.2 UInt16 SDH::cDSA::sSensorInfo::nb\_matrices**

**10.13.2.3 UInt16 SDH::cDSA::sSensorInfo::generated\_by**

**10.13.2.4 UInt8 SDH::cDSA::sSensorInfo::hw\_revision**

**10.13.2.5 UInt32 SDH::cDSA::sSensorInfo::serial\_no**

**10.13.2.6 UInt8 SDH::cDSA::sSensorInfo::feature\_flags**

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

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

## 10.14 SDH::cDSA::sTactileSensorFrame Struct Reference

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

### 10.14.1 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
- 

### Public Member Functions

- [sTactileSensorFrame \(void\)](#)

*constructor*

- [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.*

- [tTexel \\* texel](#)

*an 2D array of tTexel elements. Use [GetTexel\(\)](#) for easy access to specific individual elements.*

### 10.14.2 Constructor & Destructor Documentation

#### 10.14.2.1 SDH::cDSA::sTactileSensorFrame::sTactileSensorFrame (void) [inline]

*constructor*

**10.14.2.2 SDH::cDSA::sTactileSensorFrame::sTactileSensorFrame (void) [inline]**

constructor

### 10.14.3 Member Data Documentation

**10.14.3.1 UInt32 SDH::cDSA::sTactileSensorFrame::timestamp**

the timestamp of the frame. Use [GetAgeOfFrame\(\)](#) to set this into relation with the time of the PC.

**10.14.3.2 UInt8 SDH::cDSA::sTactileSensorFrame::flags**

internal data

**10.14.3.3 tTexel\* SDH::cDSA::sTactileSensorFrame::texel**

an 2D array of tTexel elements. Use [GetTexel\(\)](#) for easy access to specific individual elements.

**10.14.3.4 tTexel\* SDH::cDSA::sTactileSensorFrame::texel**

an 2D array of tTexel elements. Use [GetTexel\(\)](#) for easy access to specific individual elements.

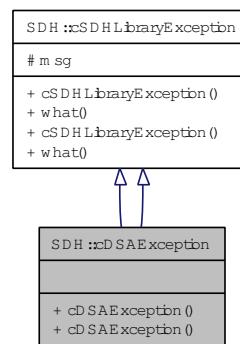
The documentation for this struct was generated from the following files:

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

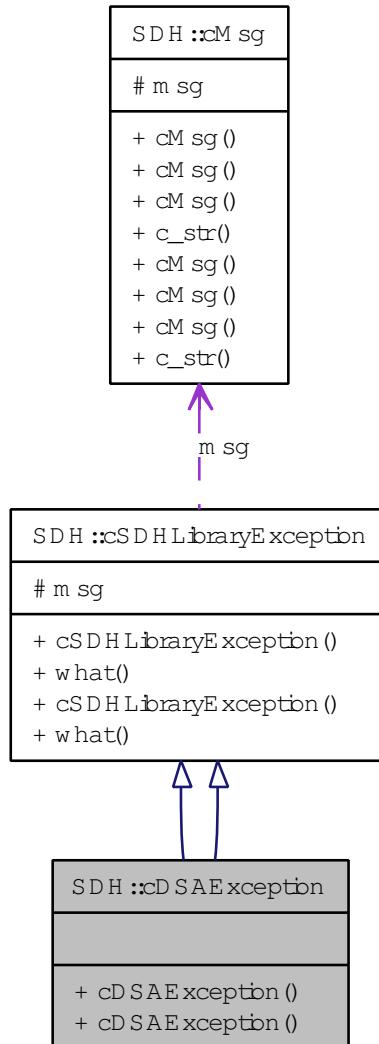
## 10.15 SDH::cDSAException Class Reference

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

Inheritance diagram for SDH::cDSAException:



Collaboration diagram for SDH::cDSAException:



### 10.15.1 Detailed Description

Derived exception class for low-level DSA related exceptions.

### Public Member Functions

- [cDSAException \(cMsg const &\\_msg\)](#)
- [cDSAException \(cMsg const &\\_msg\)](#)

## 10.15.2 Constructor & Destructor Documentation

**10.15.2.1 SDH::cDSAException::cDSAException (cMsg const & *\_msg*) [inline]**

**10.15.2.2 SDH::cDSAException::cDSAException (cMsg const & *\_msg*) [inline]**

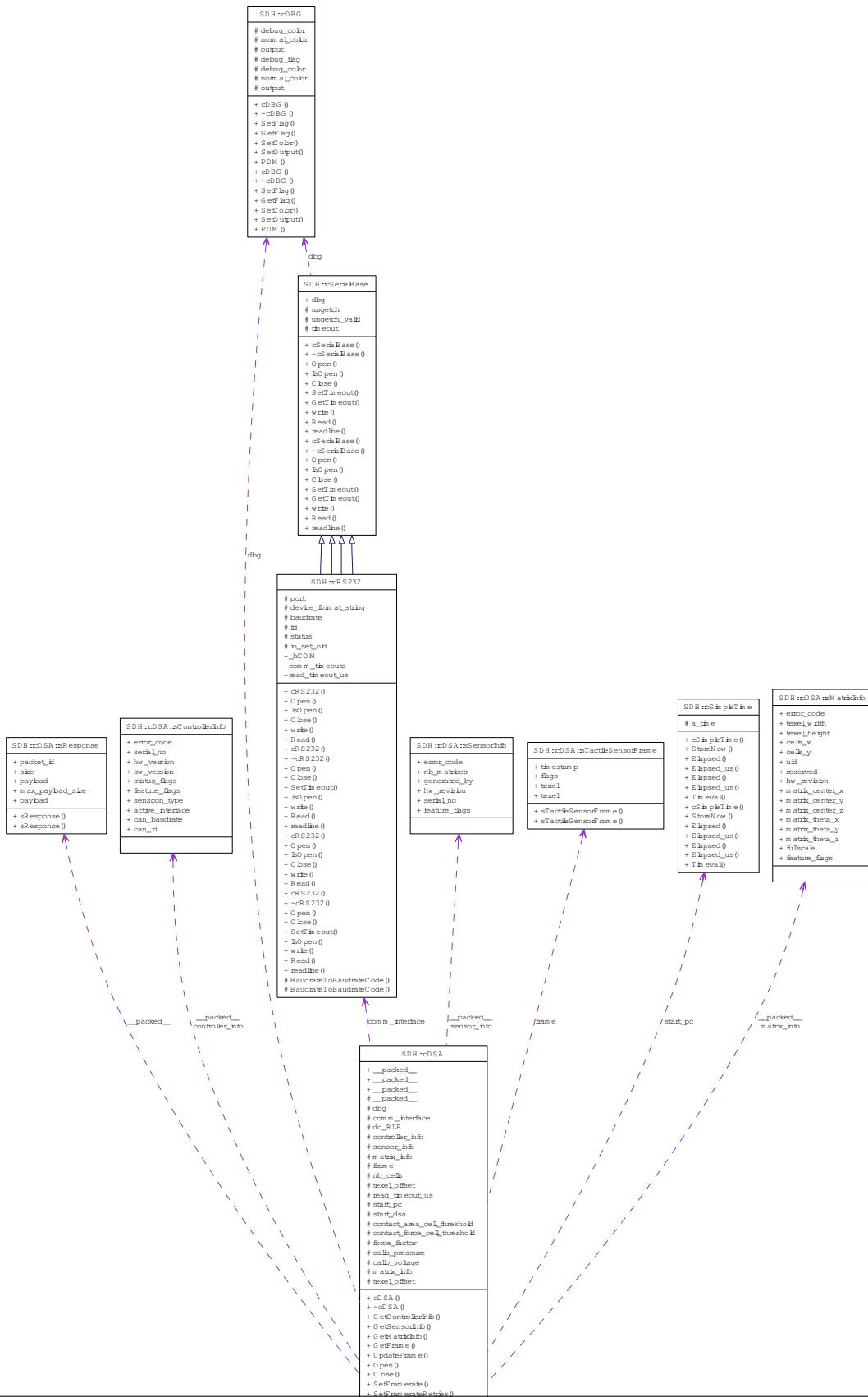
The documentation for this class was generated from the following files:

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

## 10.16 SDH::cDSAUpdater Class Reference

```
#include <dsabooost.h>
```

## Collaboration diagram for SDH::cDSAUpdater:



## Public Member Functions

- `cDSAUpdater (cDSA *_ts, int _error_threshold=DEFAULT_ERROR_THRESHOLD)`
- `void interrupt ()`  
*interrupt the updater thread*
- `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*

### 10.16.1 Constructor & Destructor Documentation

#### 10.16.1.1 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:

- \_ts* - ptr to already initialized `cDSA` tactile sensor object
- error\_threshold* - the number of errors that causes a "reset" of the connection to the remote DSA:  
communication errors with the remote DSA CON32m controllers are counted,
  - every error increments the error level,
  - every correct communication decrements the error level (down to 0) if the error level rises above `ERROR_THRESHOLD` then the connection is closed and the reopened. This is useful to recover from emergency stop since the hands loose power temporarily in that case.

#### 10.16.1.2 SDH::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:

- \_ts* - ptr to already initialized `cDSA` tactile sensor object
- error\_threshold* - the number of errors that causes a "reset" of the connection to the remote DSA:  
communication errors with the remote DSA CON32m controllers are counted,
  - every error increments the error level,
  - every correct communication decrements the error level (down to 0) if the error level rises above `ERROR_THRESHOLD` then the connection is closed and the reopened. This is useful to recover from emergency stop since the hands loose power temporarily in that case.

## 10.16.2 Member Function Documentation

### 10.16.2.1 void SDH::cDSAUpdater::interrupt () [inline]

interrupt the updater thread

### 10.16.2.2 void SDH::cDSAUpdater::interrupt () [inline]

interrupt the updater thread

## 10.16.3 Member Data Documentation

### 10.16.3.1 static int const SDH::cDSAUpdater::DEFAULT\_ERROR\_THRESHOLD = 16 [static]

default error threshold, see parameter *error\_threshold* in CTOR

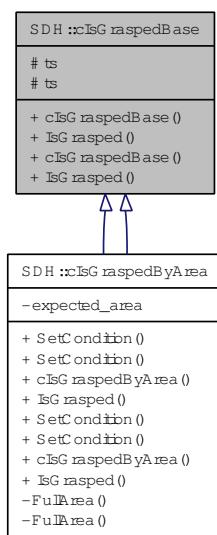
The documentation for this class was generated from the following files:

- [cpp.desire.final/demo/dsaboost.h](#)
- [demo/dsaboost.h](#)
- [cpp.desire.final/demo/dsaboost.cpp](#)
- [demo/dsaboost.cpp](#)

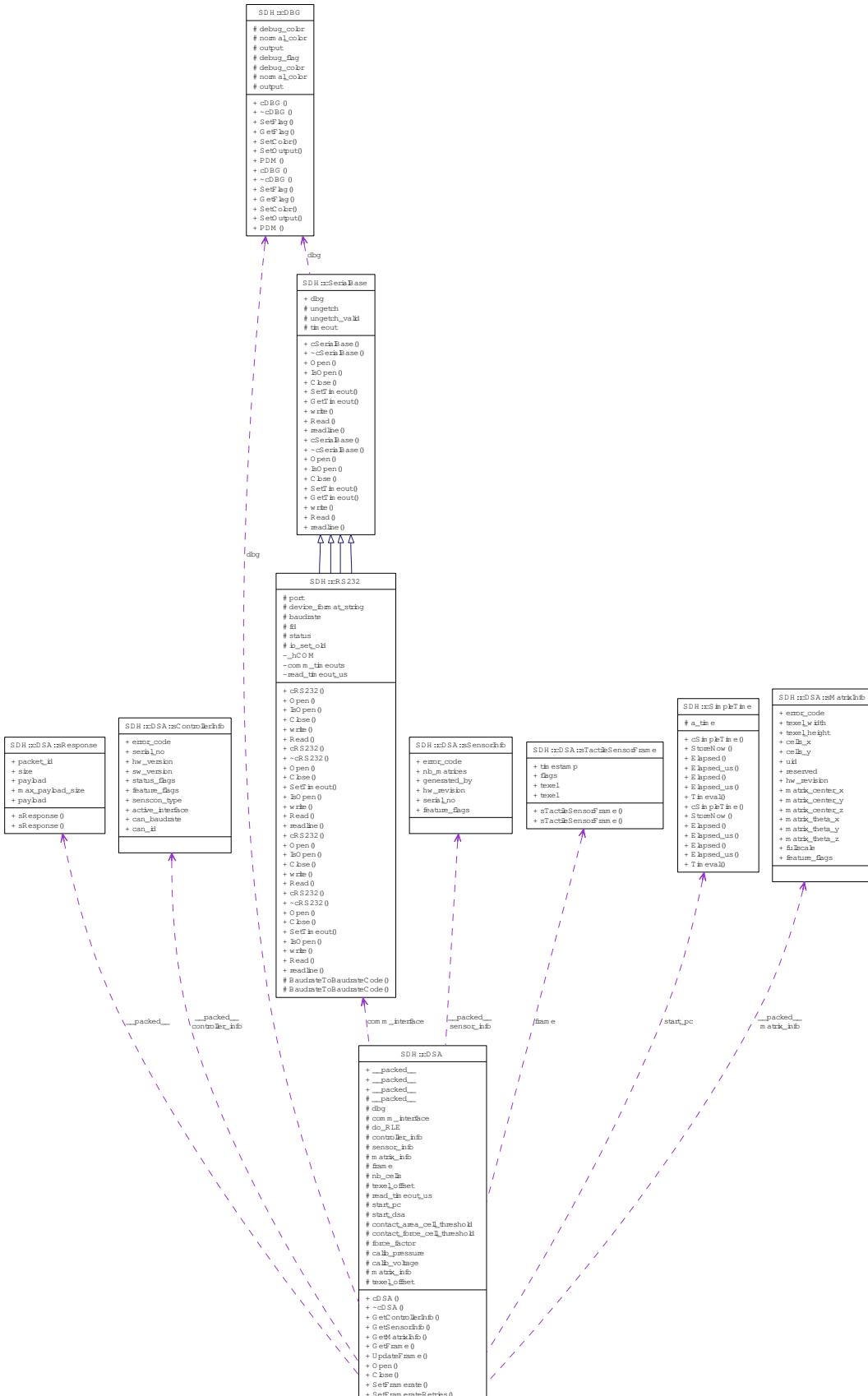
## 10.17 SDH::cIsGraspedBase Class Reference

```
#include <dsabooost.h>
```

Inheritance diagram for SDH::cIsGraspedBase:



Collaboration diagram for SDH::cIsGraspedBase:



### 10.17.1 Detailed Description

abstract base class for calculation of IsGrasped condition

#### Public Member Functions

- `cIsGraspedBase (SDH::cDSA *_ts)`
- virtual bool `IsGrasped ()=0`
- `cIsGraspedBase (SDH::cDSA *_ts)`
- virtual bool `IsGrasped ()=0`

#### Protected Attributes

- `SDH::cDSA * ts`  
*ptr to the `cDSA` tactile sensor object to use*
- `SDH::cDSA * ts`  
*ptr to the `cDSA` tactile sensor object to use*

### 10.17.2 Constructor & Destructor Documentation

**10.17.2.1 SDH::cIsGraspedBase::cIsGraspedBase (SDH::cDSA \*\_ts) [inline]**

**10.17.2.2 SDH::cIsGraspedBase::cIsGraspedBase (SDH::cDSA \*\_ts) [inline]**

#### 10.17.3 Member Function Documentation

**10.17.3.1 virtual bool SDH::cIsGraspedBase::IsGrasped (void) [pure virtual]**

Implemented in `SDH::cIsGraspedByArea`, and `SDH::cIsGraspedByArea`.

**10.17.3.2 virtual bool SDH::cIsGraspedBase::IsGrasped (void) [pure virtual]**

Implemented in `SDH::cIsGraspedByArea`, and `SDH::cIsGraspedByArea`.

#### 10.17.4 Member Data Documentation

**10.17.4.1 SDH::cDSA\* SDH::cIsGraspedBase::ts [protected]**

ptr to the `cDSA` tactile sensor object to use

**10.17.4.2 SDH::cDSA\* SDH::cIsGraspedBase::ts [protected]**

ptr to the `cDSA` tactile sensor object to use

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

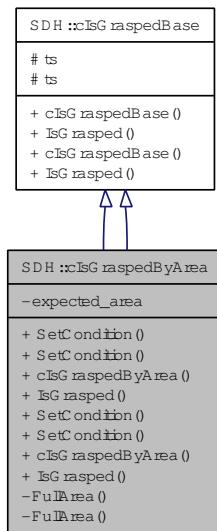
- `cpp.desire.final/demo/dsaboost.h`

- demo/[dsaboost.h](#)

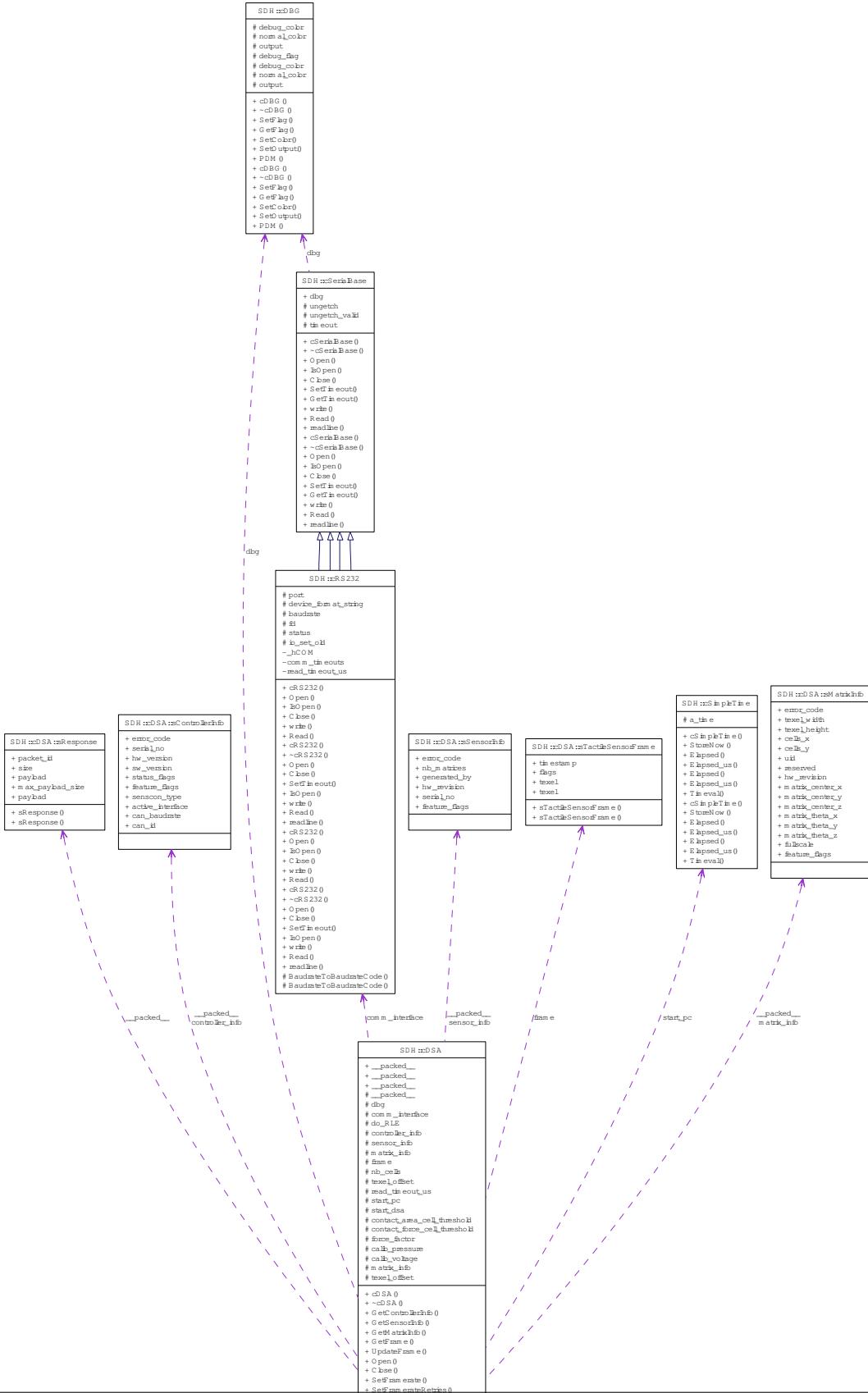
## 10.18 SDH::cIsGraspedByArea Class Reference

```
#include <dsabooost.h>
```

Inheritance diagram for SDH::cIsGraspedByArea:



Collaboration diagram for SDH::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)
- 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)

### 10.18.1 Constructor & Destructor Documentation

#### 10.18.1.1 [cIsGraspedByArea::cIsGraspedByArea](#) ([SDH::cDSA](#) \*\_ts)

default constructor which initializes the internal date

#### 10.18.1.2 [SDH::cIsGraspedByArea::cIsGraspedByArea](#) ([SDH::cDSA](#) \*\_ts)

default constructor which initializes the internal date

### 10.18.2 Member Function Documentation

#### 10.18.2.1 [void cIsGraspedByArea::SetCondition](#) (double *eaf0p*, double *eaf0d*, double *eaf1p*, double *eaf1d*, double *eaf2p*, double *eaf2d*)

set the is grasped condition

#### Parameters:

*eaf0p* - expected area of finger 0 proximal for detecting grasp condition, value [0..1] with 0=no area, 1=full area

*eaf0d* - expected area of finger 0 distal for detecting grasp condition, value [0..1] with 0=no area, 1=full area

*eaf1p* - ...

*eaf1d* - ...

*eaf2p* - ...

*eaf2d* - ...

**10.18.2.2 void cIsGraspedByArea::SetCondition (double \* *eafx*)**

overloaded variant which uses an array of doubles instead of 6 single double parameters

**10.18.2.3 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 [SDH::cIsGraspedBase](#).

**10.18.2.4 void SDH::cIsGraspedByArea::SetCondition (double *eaf0p*, double *eaf0d*, double *eaf1p*, double *eaf1d*, double *eaf2p*, double *eaf2d*)**

set the is grasped condition

**Parameters:**

*eaf0p* - expected area of finger 0 proximal for detecting grasp condition, value [0..1] with 0=no area,  
1=full area

*eaf0d* - expected area of finger 0 distal for detecting grasp condition, value [0..1] with 0=no area,  
1=full area

*eaf1p* - ...

*eaf1d* - ...

*eaf2p* - ...

*eaf2d* - ...

**10.18.2.5 void SDH::cIsGraspedByArea::SetCondition (double \* *eafx*)**

overloaded variant which uses an array of doubles instead of 6 single double parameters

**10.18.2.6 virtual bool SDH::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 [SDH::cIsGraspedBase](#).

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

- [cpp.desire.final/demo/dsaboost.h](#)
- [demo/dsaboost.h](#)
- [cpp.desire.final/demo/dsaboost.cpp](#)
- [demo/dsaboost.cpp](#)

## 10.19 SDH::cMsg Class Reference

```
#include <sdhexception.h>
```

### 10.19.1 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.

### 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.*
- [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 = 256](#) }  
*anonymous enum instead of define macros*
- enum { [eMAX\\_MSG = 256](#) }  
*anonymous enum instead of define macros*

## Protected Attributes

- char `msg` [eMAX\_MSG]

### 10.19.2 Member Enumeration Documentation

#### 10.19.2.1 anonymous enum [protected]

anonymous enum instead of define macros

##### Enumerator:

`eMAX_MSG` maximum length in bytes of a message to store

#### 10.19.2.2 anonymous enum [protected]

anonymous enum instead of define macros

##### Enumerator:

`eMAX_MSG` maximum length in bytes of a message to store

### 10.19.3 Constructor & Destructor Documentation

#### 10.19.3.1 cMsg::cMsg ()

Default constructor, init message to empty string.

#### 10.19.3.2 cMsg::cMsg (cMsg const & *other*)

Copy constructor, copy message content of other object to this object.

#### 10.19.3.3 cMsg::cMsg (char const \**fmt*, ...)

Constructor with printf like format, argument parameters.

#### 10.19.3.4 SDH::cMsg::cMsg ()

Default constructor, init message to empty string.

#### 10.19.3.5 SDH::cMsg::cMsg (cMsg const & *other*)

Copy constructor, copy message content of other object to this object.

#### 10.19.3.6 SDH::cMsg::cMsg (char const \**fmt*, ...)

Constructor with printf like format, argument parameters.

## 10.19.4 Member Function Documentation

### 10.19.4.1 `char const * cMsg::c_str () const`

Return the C-string representation of the messag in this object.

### 10.19.4.2 `char const* SDH::cMsg::c_str () const`

Return the C-string representation of the messag in this object.

## 10.19.5 Member Data Documentation

### 10.19.5.1 `char SDH::cMsg::msg [protected]`

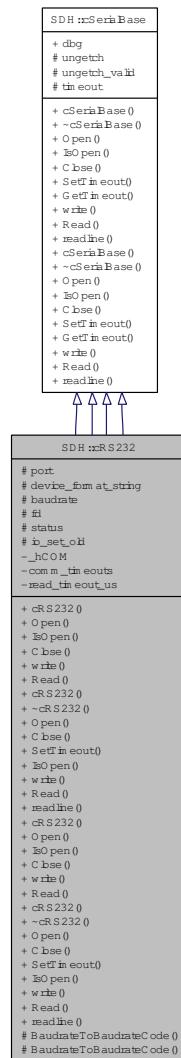
The documentation for this class was generated from the following files:

- [cpp.desire.final/sdh/sdhexception.h](#)
- [sdh/sdhexception.h](#)
- [cpp.desire.final/sdh/sdhexception.cpp](#)
- [sdh/sdhexception.cpp](#)

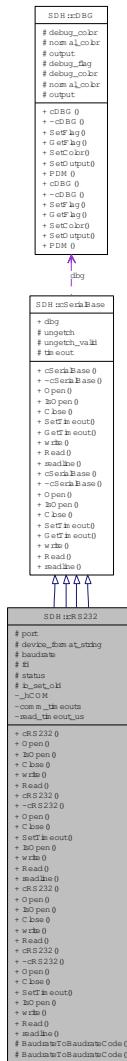
## 10.20 SDH::cRS232 Class Reference

```
#include <rs232-vcc.h>
```

Inheritance diagram for SDH::cRS232:



Collaboration diagram for SDH::cRS232:



### 10.20.1 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.

#### 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\*)
 

*(int \_port, unsigned long \_baudrate, double \_timeout, char const \*\_device\_format\_string=""")*
- **cRS232** (int \_port, unsigned long \_baudrate, double \_timeout, char const \*\_device\_format\_string="")
 

*(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 \*line, int size, char \*eol, bool return\_on\_less\_data) throw (cRS232Exception\*)*
- **cRS232** (int \_port, unsigned long \_baudrate, double \_timeout, char const \*\_device\_format\_string="/dev/ttyS%d")
 

*(void)*
- 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\*)
 

*(int \_port, unsigned long \_baudrate, double \_timeout, char const \*\_device\_format\_string=""")*
- **cRS232** (int \_port, unsigned long \_baudrate, double \_timeout, char const \*\_device\_format\_string="")
 

*(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\*)

## 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.*
- 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**

### 10.20.2 Constructor & Destructor Documentation

#### 10.20.2.1 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:

*\_port* - rs232 device number: 0='COM1'='/dev/ttys0', 1='COM2'='/dev/ttys1', ...

*\_baudrate* - the baudrate in bit/s

*\_timeout* - the timeout in seconds

*\_device\_format\_string* - 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

#### 10.20.2.2 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:**

*\_port* - rs232 device number: 0='COM1'='/dev/ttyS0', 1='COM2'='/dev/ttyS1', ...

*\_baudrate* - the baudrate in bit/s

*\_timeout* - the timeout in seconds

*\_device\_format\_string* - ignored for this VCC version

#### 10.20.2.3 cRS232::~cRS232 (void)

#### 10.20.2.4 SDH::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:**

*\_port* - rs232 device number: 0='COM1'='/dev/ttyS0', 1='COM2'='/dev/ttyS1', ...

*\_baudrate* - the baudrate in bit/s

*\_timeout* - the timeout in seconds

*\_device\_format\_string* - 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

#### 10.20.2.5 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:**

*\_port* - rs232 device number: 0='COM1'='/dev/ttyS0', 1='COM2'='/dev/ttyS1', ...

*\_baudrate* - the baudrate in bit/s

*\_timeout* - the timeout in seconds

*\_device\_format\_string* - ignored for this VCC version

### 10.20.2.6 SDH::cRS232::~cRS232 (void)

#### 10.20.3 Member Function Documentation

##### 10.20.3.1 tcflag\_t cRS232::BaudrateToBaudrateCode (unsigned long *baudrate*) throw (cRS232Exception\*) [protected]

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

##### 10.20.3.2 void cRS232::Open (void) throw (cRS232Exception\*) [virtual]

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

Implements [SDH::cSerialBase](#).

##### 10.20.3.3 bool cRS232::IsOpen (void) throw () [virtual]

Return true if port to RS232 is open.

Implements [SDH::cSerialBase](#).

##### 10.20.3.4 void cRS232::Close (void) throw (cRS232Exception\*) [virtual]

Close the previously opened rs232 port.

Implements [SDH::cSerialBase](#).

##### 10.20.3.5 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:

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

#### Returns:

the number of bytes actually written

!! dwWritten is always 0! Damn bloody windows

Implements [SDH::cSerialBase](#).

##### 10.20.3.6 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](#).

#### **10.20.3.7 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](#).

#### **10.20.3.8 void SDH::cRS232::Close (void) throw (cRS232Exception\*) [virtual]**

Close the previously opened communication channel.

Implements [SDH::cSerialBase](#).

#### **10.20.3.9 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](#).

#### **10.20.3.10 bool SDH::cRS232::IsOpen () throw () [inline, virtual]**

Return true if communication channel is open.

Implements [SDH::cSerialBase](#).

#### **10.20.3.11 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:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

##### **Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

#### **10.20.3.12 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](#).

**10.20.3.13** `char * cRS232::readline (char * line, int size, char * eol, bool return_on_less_data) throw (cRS232Exception*)`

**10.20.3.14** `tcflag_t SDH::cRS232::BaudrateToBaudrateCode (unsigned long baudrate) throw (cRS232Exception*) [protected]`

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

**10.20.3.15** `void SDH::cRS232::Open (void) throw (cRS232Exception*) [virtual]`

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

Implements [SDH::cSerialBase](#).

**10.20.3.16** `bool SDH::cRS232::IsOpen (void) throw () [virtual]`

Return true if port to RS232 is open.

Implements [SDH::cSerialBase](#).

**10.20.3.17** `void SDH::cRS232::Close (void) throw (cRS232Exception*) [virtual]`

Close the previously opened rs232 port.

Implements [SDH::cSerialBase](#).

**10.20.3.18** `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:

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

#### Returns:

the number of bytes actually written

Implements [SDH::cSerialBase](#).

---

**10.20.3.19** `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](#).

**10.20.3.20** `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](#).

**10.20.3.21** `void SDH::cRS232::Close (void) throw (cRS232Exception*) [virtual]`

Close the previously opened communication channel.

Implements [SDH::cSerialBase](#).

**10.20.3.22** `virtual void SDH::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](#).

**10.20.3.23** `bool SDH::cRS232::IsOpen () throw () [inline, virtual]`

Return true if communication channel is open.

Implements [SDH::cSerialBase](#).

**10.20.3.24** `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:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implements [SDH::cSerialBase](#).

---

**10.20.3.25** `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](#).

**10.20.3.26** `char* SDH::cRS232::readline (char * line, int size, char * eol, bool return_on_less_data) throw (cRS232Exception*)`

## 10.20.4 Member Data Documentation

**10.20.4.1** `int SDH::cRS232::port [protected]`

the RS232 portnumber to use

the RS232 port number to use (port 0 is COM1)

**10.20.4.2** `std::string SDH::cRS232::device_format_string [protected]`

the sprintf format string to generate the device name from the port, see Constructor

**10.20.4.3** `unsigned long SDH::cRS232::baudrate [protected]`

the baudrate in bit/s

**10.20.4.4** `int SDH::cRS232::fd [protected]`

the file descriptor of the RS232 port

**10.20.4.5** `int SDH::cRS232::status [protected]`
**10.20.4.6** `termios SDH::cRS232::io_set_old [protected]`

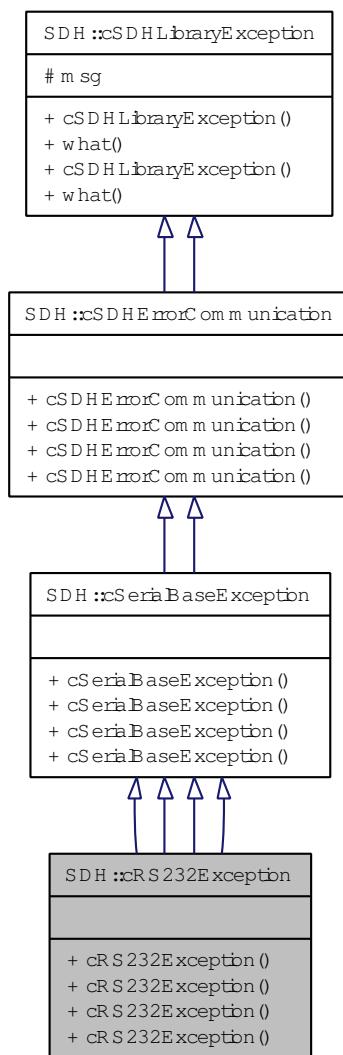
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[rs232-cygwin.h](#)
- cpp.desire.final/sdh/[rs232-vcc.h](#)
- sdh/[rs232-cygwin.h](#)
- sdh/[rs232-vcc.h](#)
- cpp.desire.final/sdh/[rs232-cygwin.cpp](#)
- cpp.desire.final/sdh/[rs232-vcc.cpp](#)
- sdh/[rs232-cygwin.cpp](#)
- sdh/[rs232-vcc.cpp](#)

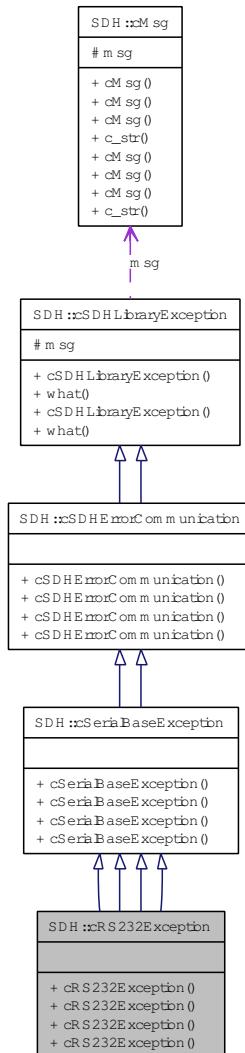
## 10.21 SDH::cRS232Exception Class Reference

```
#include <rs232-vcc.h>
```

Inheritance diagram for SDH::cRS232Exception:



Collaboration diagram for SDH::cRS232Exception:



### 10.21.1 Detailed Description

Derived exception class for low-level RS232 related exceptions.

#### Public Member Functions

- `cRS232Exception (cMsg const &_msg)`

## 10.21.2 Constructor & Destructor Documentation

**10.21.2.1 SDH::cRS232Exception::cRS232Exception (cMsg const & *\_msg*) [inline]**

**10.21.2.2 SDH::cRS232Exception::cRS232Exception (cMsg const & *\_msg*) [inline]**

**10.21.2.3 SDH::cRS232Exception::cRS232Exception (cMsg const & *\_msg*) [inline]**

**10.21.2.4 SDH::cRS232Exception::cRS232Exception (cMsg const & *\_msg*) [inline]**

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

- [cpp.desire.final/sdh/rs232-cygwin.h](#)
- [cpp.desire.final/sdh/rs232-vcc.h](#)
- [sdh/rs232-cygwin.h](#)
- [sdh/rs232-vcc.h](#)

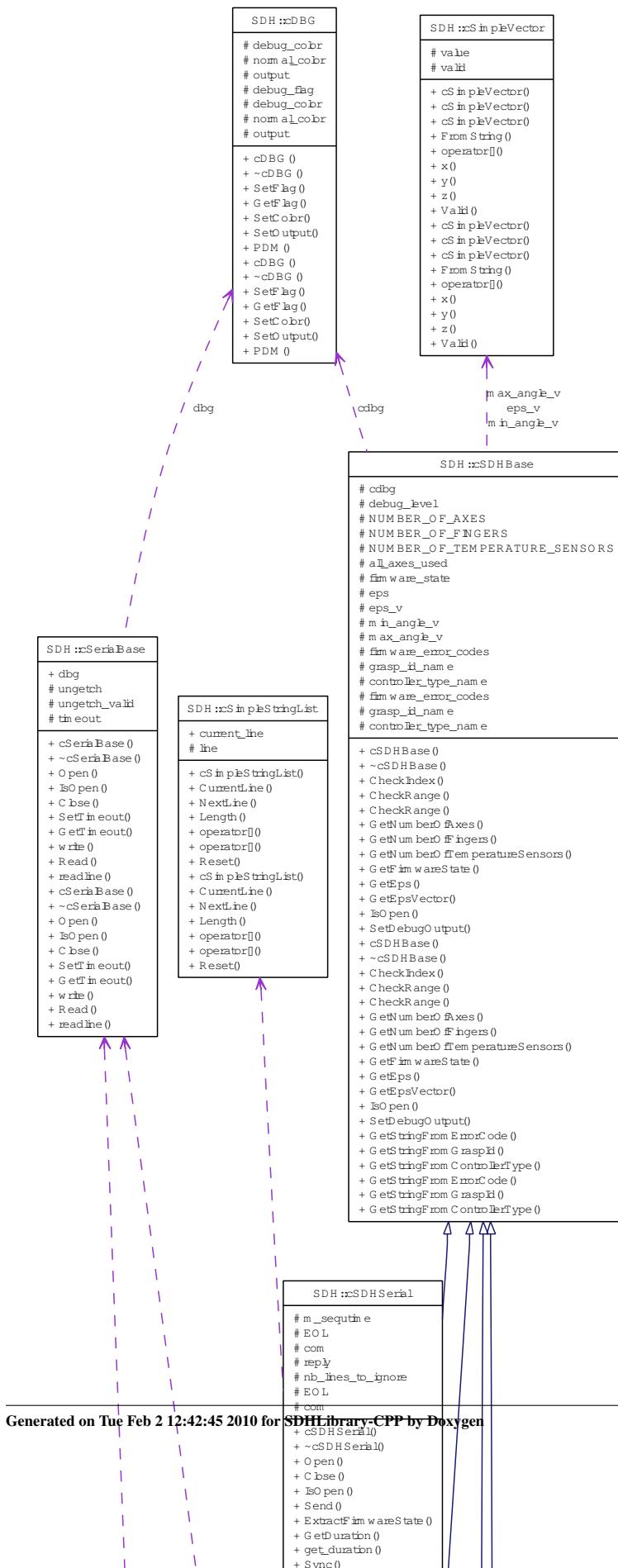
## 10.22 SDH::cSDH Class Reference

```
#include <sdh.h>
```

Inheritance diagram for SDH::cSDH:



## Collaboration diagram for SDH::cSDH:



### 10.22.1 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 parmeteres: 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 refered 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 (USER-APP and SDHLibrary-CPP in the [overview](#) figure"). For now the SDH firmware knows only about its internal unit system.

### 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.*

- virtual void **SetDebugOutput** (std::ostream \*debuglog)  
*change the stream to use for debug messages*

## Kinematic parameters of the Hand

- std::vector< std::vector< double > > **offset**
- **cSerialBase** \* **com**
- 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\*)
- 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)

## 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\*)
- 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)

## Public Types

- enum `eMotorCurrentMode` {
   
`eMCM_MOVE` = 0, `eMCM_GRIP` = 1, `eMCM_HOLD` = 2, `eMCM_DIMENSION`,  
`eMCM_MOVE` = 0, `eMCM_GRIP` = 1, `eMCM_HOLD` = 2, `eMCM_DIMENSION` }
   
*the motor current can be set specifically for these modes:*
- 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`, `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 state of an axis (see TPOSCON\_STATE in global.h of the SDH firmware).*
- enum `eMotorCurrentMode` {
   
`eMCM_MOVE` = 0, `eMCM_GRIP` = 1, `eMCM_HOLD` = 2, `eMCM_DIMENSION`,  
`eMCM_MOVE` = 0, `eMCM_GRIP` = 1, `eMCM_HOLD` = 2, `eMCM_DIMENSION` }
   
*the motor current can be set specifically for these modes:*
- 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`, `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 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)
   
*Constructor of `cSDH` class.*
- virtual `~cSDH` ()
   
`cSDH` (bool \_use\_radians=false, bool \_use\_fahrenheit=false, int \_debug\_level=0)
   
*Constructor of `cSDH` class.*
- virtual `~cSDH` ()

## 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\_t \_id\_read=43, int32\_t \_id\_write=42) throw (cSDHLibraryException\*)
- void `OpenCAN_ESD` (NTCAN\_HANDLE \_ntcan\_handle, double \_timeout=0.0, int32\_t \_id\_read=43, int32\_t \_id\_write=42) throw (cSDHLibraryException\*)
- void `OpenCAN_Peak` (unsigned long \_baudrate=1000000, double \_timeout=0.0, int32\_t \_id\_read=43, int32\_t \_id\_write=42, const char \*device="/dev/pcanusb0") throw (cSDHLibraryException\*)
- void `OpenCAN_Peak` (HANDLE \_handle, double \_timeout=0.0, int32\_t \_id\_read=43, int32\_t \_id\_write=42) 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 > `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 > `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\*)

### 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` (NTCAN\_HANDLE \_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` (PCAN\_HANDLE \_handle, double \_timeout=0.0, Int32 \_id\_read=43, Int32 \_id\_write=42) 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 > **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 > **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*

### 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 objecs

*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 accelerations: 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_millampere`  
*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

- void `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*)`

### Internal helper methods

- `void 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)).*
- 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.*

## 10.22.2 Member Enumeration Documentation

### 10.22.2.1 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.
- 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.

### 10.22.2.2 enum SDH::cSDH::eAxisState

The state of an axis (see TPOSSTATE 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 in against counterwise direction  
**eAS\_DISABLED** axis is disabled  
**eAS\_LIMITS\_REACHED** position limits reached and axis stopped  
**eAS\_DIMENSION** Endmarker and Dimension.  
**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 in against counterwise direction  
**eAS\_DISABLED** axis is disabled  
**eAS\_LIMITS\_REACHED** position limits reached and axis stopped  
**eAS\_DIMENSION** Endmarker and Dimension.

#### 10.22.2.3 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.  
**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.

#### 10.22.2.4 enum SDH::cSDH::eAxisState

The state of an axis (see TPOSSTATE 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 in against counterwise direction  
*eAS\_DISABLED* axis is disabled  
*eAS\_LIMITS\_REACHED* position limits reached and axis stopped  
*eAS\_DIMENSION* Endmarker and Dimension.  
*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 in against counterwise direction  
*eAS\_DISABLED* axis is disabled  
*eAS\_LIMITS\_REACHED* position limits reached and axis stopped  
*eAS\_DIMENSION* Endmarker and Dimension.

### 10.22.3 Constructor & Destructor Documentation

#### 10.22.3.1 cSDH::cSDH (*bool \_use\_radians = false, bool \_use\_fahrenheit = false, int \_debug\_level = 0*)

Constructor of [cSDH](#) class.

Creates a 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 to 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 [°] for (axis) angles
- degrees per second [°/s] for (axis) angular velocities
- seconds [s] for times
- degrees celsius [°C] for temperatures

#### Parameters:

- *\_use\_radians* : Flag, if true then use radians and radians/second to set/report (axis) angles and angular velocities instead of default degrees and degrees/s.
- *\_use\_fahrenheit* : Flag, if true then use degrees fahrenheit to report temperatures instead of default degrees celsius.
- *\_debug\_level* : The level of debug messages to print
  - 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))

### 10.22.3.2 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).

### 10.22.3.3 SDH::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 [ $^{\circ}$ ] for (axis) angles
- degrees per second [ $^{\circ}/s$ ] for (axis) angular velocities
- seconds [s] for times
- degrees celsius [ $^{\circ}\text{C}$ ] for temperatures

#### Parameters:

*\_use\_radians* : Flag, if true then use radians and radians/second to set/report (axis) angles and angular velocities instead of default degrees and degrees/s.

*\_use\_fahrenheit* : Flag, if true then use degrees fahrenheit to report temperatures instead of default degrees celsius.

*\_debug\_level* : The level of debug messages to print

- 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 );
```

#### 10.22.3.4 virtual SDH::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).

### 10.22.4 Member Function Documentation

#### 10.22.4.1 virtual void SDH::cSDH::SetDebugOutput (std::ostream \* *debuglog*) [inline, virtual]

change the stream to use for debug messages

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

#### 10.22.4.2 void 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:**

*axes* - a vector of axis indices  
*values* - a vector of values  
*ll\_set* - a pointer to the low level set function to use  
*ll\_get* - a pointer to the low level get function to use (for those axes where the given value is NaN)  
*uc* - a pointer to the unit converter object to use before sending values to *ll\_set*  
*min\_values* - a vector with the minimum allowed values  
*max\_values* - a vector with the maximum allowed values  
*name* - a string with the name of the values (for constructing error message)

**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.

#### 10.22.4.3 `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:**

*axes* - a vector of axis indices  
*ll\_get* - a pointer to the low level get function to use  
*uc* - a pointer to the unit converter object to apply before returning values  
*name* - 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*).

---

**10.22.4.4 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:**

*index* - The index to vectorize or All

*all\_replacement* - a vector to return if *index* is All

*maxindex* - the *index* is checked if in [0.. *maxindex*] (i.e. not including *maxindex*)

*name* - 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  $\geq 0$  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.

**10.22.4.5 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*.

**10.22.4.6 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)

**10.22.4.7 bool cSDH::IsVirtualAxis (int *iAxis*) throw (cSDHLibraryException\*)**

Return true if index *iAxis* refers to a virtual axis.

**10.22.4.8 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();
```

**10.22.4.9 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
```

**10.22.4.10 int cSDH::GetFingerNumberOfAxes (int *iFinger*) throw (cSDHLibraryException\*)**

Return the number of real axes of finger with index *iFinger*.

**Parameters:**

*iFinger* - 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";
```

**10.22.4.11 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:**

*iFinger* - index of finger in range [0..NUMBER\_OF\_FINGERS-1]

*iFingerAxis* - 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 )
```

**10.22.4.12 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";
```

**10.22.4.13 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";
```

**10.22.4.14 char const \* cSDH::GetFirmwareRelease (void) throw (cSDHLibraryException\*)**

Return the 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";
```

**10.22.4.15 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) # - "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-firmware" ) )  
#      #  
# #
```

#### 10.22.4.16 `std::vector< double > cSDH::GetTemperature (std::vector< int > const & sensors) throw (cSDHLibraryException*)`

Return temperature(s) measured within the SDH.

##### Parameters:

*sensors* - A vector of indices of temperature sensors to access.

- 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}
```

#### 10.22.4.17 `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.

---

**10.22.4.18 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:**

*\_port* : 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.

*\_baudrate*,: the baudrate in bit/s, the default is 115200 which happens to be the default for the SDH too

*\_timeout* : The timeout to use:

- -1 : wait forever
- T : wait for T seconds

*\_device\_format\_string* : 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/ttyUSB3 for a third hand "hand3":
cSDH hand3();
hand3.OpenRS232( 3, 115200, -1, "/dev/ttyUSB%d" );
```

---

**10.22.4.19 void cSDH::OpenCAN\_ESD (int *\_net* = 0, unsigned long *\_baudrate* = 1000000, double *\_timeout* = 0.0, int32\_t *\_id\_read* = 43, int32\_t *\_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:**

*\_net* : The ESD CAN net number of the CAN port to use. (default: 0)

*\_baudrate* : the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default),800000,500000,250000,125000,100000,50000,20000,10000)

*\_timeout* : The timeout to use:

- <= 0 : wait forever (default)
- T : wait for T seconds

*\_id\_read* - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)

*\_id\_write* - 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 );
```

**10.22.4.20 void cSDH::OpenCAN\_ESD (NTCAN\_HANDLE *\_ntcan\_handle*, double *\_timeout* = 0.0, int32\_t *\_id\_read* = 43, int32\_t *\_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:**

- \_ntcan\_handle* : The ESD CAN handle to reuse to connect to the ESD CAN driver
- \_timeout* : The timeout to use:
  - <= 0 : wait forever (default)
  - T : wait for T seconds
- \_id\_read* - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)
- \_id\_write* - 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 ESD NTCAN_HANDLE

// use default parameters for timeout and IDs
hand.OpenCAN_ESD( handle );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_ESD( handle, 1.0, 0x143, 0x142 );
```

**10.22.4.21 void cSDH::OpenCAN\_PEAK (unsigned long *\_baudrate* = 1000000, double *\_timeout* = 0.0, int32\_t *\_id\_read* = 43, int32\_t *\_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:**

- \_baudrate* : the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default), 800000, 500000, 250000, 125000, 100000, 50000, 20000, 10000)
- \_timeout* : The timeout to use:
  - <= 0 : wait forever (default)

- T : wait for T seconds

*\_id\_read* - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)  
*\_id\_write* - the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x02a)  
*\_device* - 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/pc
hand.OpenCAN_Peak( 500000, 1.0, 0x143, 0x142, "/dev/pcanusb1" );
```

#### 10.22.4.22 void cSDH::OpenCAN\_Peak (HANDLE *\_handle*, double *\_timeout* = 0.0, int32\_t *\_id\_read* = 43, int32\_t *\_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:

*\_handle* : The PEAK CAN handle to reuse to connect to the PEAK CAN driver  
*\_timeout* : The timeout to use:

- <= 0 : wait forever (default)
- T : wait for T seconds

*\_id\_read* - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)  
*\_id\_write* - the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x2a)  
*\_device* - the PEAK device name. Used for the Linux char dev driver only. default="/dev/pcanusb0"

#### Examples:

```
// Assuming 'hand' is a cSDH object ...
// and 'handle' is a valid PEAK NTCAN_HANDLE

// use default parameters for timeout and IDs
hand.OpenCAN_Peak( handle );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_Peak( handle, 1.0, 0x143, 0x142 );
```

#### 10.22.4.23 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:**

*leave\_enabled* - 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 );
```

**10.22.4.24 bool cSDH::IsOpen (void) throw () [virtual]**

Return true if connection to SDH firmware/hardware is open.

Implements [SDH::cSDHBase](#).

**10.22.4.25 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();
```

**10.22.4.26 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();
```

### **10.22.4.27 void cSDH::SetController (cSDHBase::eControllerType *controller*) throw (cSDHLIBRARYException\*)**

Set the type of axis controller to be used in the SDH

With SDH firmware >= 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 <= 0.0.2.5: only eCT\_POSE
- firmware >= 0.0.2.6: eCT\_POSE, eCT\_VELOCITY, eCT\_VELOCITY\_ACCELERATION

## Parameters:

***controller*** - 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 );
```

**10.22.4.28 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";
```

**10.22.4.29 void cSDH::SetVelocityProfile (eVelocityProfile *velocity\_profile*) throw  
(cSDHLibraryException\*)**

Set the type of velocity profile to be used in the SDH

**Parameters:**

*velocity\_profile* - 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 )
```

**10.22.4.30 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
```

### **10.22.4.31 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:**

*axes* - A vector of axis indices to access.

*motor\_currents* - 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](#).

*mode* - 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 );

```

#### 10.22.4.32 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.

#### 10.22.4.33 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:

- axes* - A vector of axis indices to access.
- mode* - 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};
```

#### **10.22.4.34 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.

#### **10.22.4.35 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 off when not needed.

#### Parameters:

`axes` - A vector of axis indices to access.  
`states` - 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 );
```

#### 10.22.4.36 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.

#### 10.22.4.37 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.

#### 10.22.4.38 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.

#### 10.22.4.39 `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:**

*axes* - 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}

// 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
```

#### 10.22.4.40 `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.

#### 10.22.4.41 `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:**

*axes* - 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_IDLE, 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}
```

#### **10.22.4.42 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.

#### **10.22.4.43 void cSDH::WaitAxis (std::vector< int > const & *axes*, double *timeout* = -1.0) throw (cSDHLibraryException\*)**

Wait until the movement(s) of of axis(axes) has finished

The state of the given axis(axes) is(are) queried until all axes are no longer moving.

**Parameters:**

- axes* - A vector of axis indices to access.  
*timeout* - 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 shure 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

// The previous command returned immediately, so
// before doing something else with the hand make shure the
// selected axes have stopped:
hand.WaitAxis( axes123 );

// now the axes have stopped
```

#### 10.22.4.44 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 axes.

#### 10.22.4.45 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:**

*axes* - A vector of axis indices to access.

*angles* - 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 [0 .. [f\\_max\\_angle\\_v](#)], i.e. it is checked that *angles[i]*, converted to internal units, is in [0 .. [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°:
hand.SetAxisTargetAngle( 3, -42.0 );

// Set target angle of for axis 0, 4 and 2 to 0.0°, -44.4° and -2.22° 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° (home-position)
hand.SetAxisTargetAngle( hand.All, 0.0 );
```

#### 10.22.4.46 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.

#### 10.22.4.47 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:

*axes* - 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}
```

#### 10.22.4.48 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.

#### 10.22.4.49 `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:**

*axes* - 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}
```

#### 10.22.4.50 `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.

#### 10.22.4.51 `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. This means that in eCT\_VELOCITY\_ACCELERATION controller type the accelerations must be set with [SetAxisTargetAcceleration\(\)](#) before calling [SetAxisTargetVelocity\(\)](#).

**Parameters:**

*axes* - A vector of axis indices to access.

*velocities* - 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 [uc\\_angular\\_velocity](#).

**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 [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\[i\]\]](#)].
- 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°/s:
hand.SetAxisTargetVelocity( 3, 42.0 );

// Set target velocity of for axis 0,4 and 2 to 0.0°/s, 44.4°/s and 2.22°/s 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°/s
hand.SetAxisTargetVelocity( hand.All, 47.11 );

```

#### **10.22.4.52 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.

#### **10.22.4.53 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:**

*axes* - 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}

```

**10.22.4.54 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.

**10.22.4.55 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:**

*axes* - 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, 120.0}

// 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}
```

**10.22.4.56 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.

#### 10.22.4.57 `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:

*axes* - 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, 120.0}

// 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}
```

#### 10.22.4.58 `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.

#### 10.22.4.59 `std::vector< double > cSDH::GetAxisActualVelocity (std::vector< int > const & axes) throw (cSDHLibraryException*)`

Get the actual velocity(s) of axis(axes).

**Parameters:**

*axes* - 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
```

**10.22.4.60 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.

**10.22.4.61 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:**

*axes* - 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.

**10.22.4.62 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.

**10.22.4.63 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:

*axes* - A vector of axis indices to access.

*accelerations* - 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°/s²:
hand.SetAxisTargetAcceleration( 3, 420.0 );

// Set target acceleration of for axis 0,4 and 2 to 0.0°/s², 444.0°/s² and 222°/s² 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°/s
hand.SetAxisTargetAcceleration( hand.All, 142.1 );

```

#### **10.22.4.64 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.

#### **10.22.4.65 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:**

*axes* - 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, 100.0}

// 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}
```

**10.22.4.66 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.

**10.22.4.67 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:**

*axes* - 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}

// 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, -1.5707963267948966, -1.5707963267948966}
```

#### 10.22.4.68 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.

#### 10.22.4.69 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:

*axes* - 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 }
```

#### 10.22.4.70 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.

#### 10.22.4.71 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:

*axes* - 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}
```

#### 10.22.4.72 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.

#### 10.22.4.73 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:

*axes* - 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, 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}
```

#### 10.22.4.74 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.

#### 10.22.4.75 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:

*axes* - A vector of axis indices to access.

*sequ* - 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 addressing 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°. The
// target poses for axes 4 and 2 are still set since the
// last MoveAxes() call was sequentially (and 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 (and thus it could 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**

**10.22.4.76 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).

**10.22.4.77 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 off when not needed.

**Parameters:**

*fingers* - A vector of finger indices to access.

*states* - 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 );
```

#### **10.22.4.78 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.

#### **10.22.4.79 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.

#### **10.22.4.80 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.

#### **10.22.4.81 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:**

*fingers* - 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
```

#### 10.22.4.82 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

#### 10.22.4.83 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:

*iFinger* - index of finger to access. This must be a single index.  
*angles* - the angle(s) to set or None 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° }
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° }
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°
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!

```

#### **10.22.4.84 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*.

#### **10.22.4.85 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:**

*iFinger* - 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.

```

**10.22.4.86 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.

**10.22.4.87 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:**

*iFinger* - 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.
```

**10.22.4.88 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.

**10.22.4.89 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:**

*iFinger* - 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}
```

#### 10.22.4.90 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.

#### 10.22.4.91 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:**

*iFinger* - 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}
```

**10.22.4.92 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.

**10.22.4.93 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:**

*iFinger* - index of finger to access. This must be a single index  
*angles* - 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, 32.600000000000001, 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, 119.60000000000002, -53.0}

// Or if you change the angle unit system:
hand.UseRadians();
position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.5707963267948966, -1.5707963267948966
// now position is still something like {18.821618775581804, 119.60000000000002, -53.0}

// Or if you change the position unit system too:
hand.uc_position = &cSDH::uc_position_meter
position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.5707963267948966, -1.5707963267948966
// now position is still something like {0.018821618775581, 0.119.60000000000002, -0.05299999}
```

#### **10.22.4.94 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*.

#### **10.22.4.95 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:**

*fingers* - A vector of finger indices to access.  
*sequ* - 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 finger 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 overwritten
// 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**

#### **10.22.4.96 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).

#### **10.22.4.97 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**

#### **10.22.4.98 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
```

#### 10.22.4.99 double cSDH::GripHand (eGraspId *grip*, double *close*, double *velocity*, bool *sequ* = 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 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. 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 [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:**

*grip* - The index of the grip to perform [0..eGID\_DIMENSION-1] (s.a. [eGraspId](#))

*close* - close-ratio: [0.0 .. 1.0] where 0.0 is 'fully opened' and 1.0 is 'fully closed'

*velocity* - maximum allowed angular axis velocity in the chosen external unit system [uc\\_angular\\_velocity](#)

*sequ* - 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 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.
- 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 centrical grip at 50°/s:
hand.GripHand( hand.eGID_CENTRICAL, 0.0, 50.0, true );

// Now close it 50% with 30°/s:
hand.GripHand( hand.eGID_CENTRICAL, 0.5, 30.0, true );

// Then close it completely with 20°/s:
hand.GripHand( hand.eGID_CENTRICAL, 1.0, 20.0, true );

```

#### **10.22.4.100 virtual void SDH::cSDH::SetDebugOutput (std::ostream \* *debuglog*) [inline, virtual]**

change the stream to use for debug messages

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

#### **10.22.4.101 void SDH::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:**

*axes* - a vector of axis indices

*values* - a vector of values

*ll\_set* - a pointer to the low level set function to use

*ll\_get* - a pointer to the low level get function to use (for those axes where the given value is NaN)

*uc* - a pointer to the unit converter object to use before sending values to *ll\_set*

*min\_values* - a vector with the minimum allowed values

*max\_values* - a vector with the maximum allowed values

*name* - a string with the name of the values (for constructing error message)

##### **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* .. *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.

---

**10.22.4.102 std::vector<double> SDH::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:**

*axes* - a vector of axis indices  
*ll\_get* - a pointer to the low level get function to use  
*uc* - a pointer to the unit converter object to apply before returning values  
*name* - 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*).

**10.22.4.103 std::vector<int> SDH::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:**

*index* - The index to vectorize or [All](#)  
*all\_replacement* - a vector to return if *index* is [All](#)  
*maxindex* - the *index* is checked if in [0.. *maxindex*] (i.e. not including *maxindex*)  
*name* - 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  $\geq 0$  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.

**10.22.4.104 pSetFunction SDH::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*.

---

**10.22.4.105 std::vector<double> SDH::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)

**10.22.4.106 bool SDH::cSDH::IsVirtualAxis (int *iAxis*) throw (cSDHLibraryException\*)**

Return true if index *iAxis* refers to a virtual axis.

**10.22.4.107 void SDH::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();
```

**10.22.4.108 void SDH::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
```

**10.22.4.109 int SDH::cSDH::GetFingerNumberOfAxes (int *iFinger*) throw (cSDHLibraryException\*)**

Return the number of real axes of finger with index *iFinger*.

**Parameters:**

*iFinger* - 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";
```

**10.22.4.110 int SDH::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:**

*iFinger* - index of finger in range [0..NUMBER\_OF\_FINGERS-1]

*iFingerAxis* - 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 )
```

**10.22.4.111 static char const\* SDH::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";
```

**10.22.4.112 static char const\* SDH::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";
```

#### 10.22.4.113 `char const* SDH::cSDH::GetFirmwareRelease (void) throw (cSDHLibraryException*)`

Return the 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";
```

#### 10.22.4.114 `char const* SDH::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) # - "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-firmware" ) )
# #

# #
```

#### 10.22.4.115 `std::vector<double> SDH::cSDH::GetTemperature (std::vector< int > const & sensors) throw (cSDHLibraryException*)`

Return temperature(s) measured within the SDH.

##### Parameters:

`sensors` - A vector of indices of temperature sensors to access.

- 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, 52.000 }

// 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}
```

**10.22.4.116 double SDH::cSDH::GetTemperature (int *iSensor*) throw (cSDHLibraryException\*)**

Like [GetTemperature\(std::vector<int>const&\)](#), just for one sensor *iSensor* and returning a single temperature as double.

**10.22.4.117 void SDH::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:**

- \_port*** : 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.
- \_baudrate*:** the baudrate in bit/s, the default is 115200 which happens to be the default for the SDH too
- \_timeout*** : The timeout to use:
  - -1 : wait forever
  - T : wait for T seconds

***\_device\_format\_string*** : 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/ttyUSB3 for a third hand "hand3":
cSDH hand3();
hand3.OpenRS232( 3, 115200, -1, "/dev/ttyUSB%d" );
```

#### 10.22.4.118 void SDH::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:

***\_net*** : The ESD CAN net number of the CAN port to use. (default: 0)  
***\_baudrate*** : the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default), 800000, 500000, 250000, 125000, 100000, 50000, 20000, 10000)  
***\_timeout*** : The timeout to use:

- <= 0 : wait forever (default)
- T : wait for T seconds

***\_id\_read*** - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)  
***\_id\_write*** - 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 );
```

#### 10.22.4.119 void SDH::cSDH::OpenCAN\_ESD (NTCAN\_HANDLE *\_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:**

- \_ntcan\_handle** : The ESD CAN handle to reuse to connect to the ESD CAN driver
- \_timeout** : The timeout to use:
  - <= 0 : wait forever (default)
  - T : wait for T seconds
- \_id\_read** - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)
- \_id\_write** - 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 ESD NTCAN_HANDLE

// use default parameters for timeout and IDs
hand.OpenCAN_ESD( handle );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_ESD( handle, 1.0, 0x143, 0x142 );
```

#### 10.22.4.120 void SDH::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:**

- \_baudrate** : the CAN baudrate in bit/s. Only some bitrates are valid: (1000000 (default), 800000, 500000, 250000, 125000, 100000, 50000, 20000, 10000)
- \_timeout** : The timeout to use:
  - <= 0 : wait forever (default)
  - T : wait for T seconds
- \_id\_read** - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x02b)
- \_id\_write** - the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x02a)
- \_device** - 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/pca
hand.OpenCAN_PEAK( 500000, 1.0, 0x143, 0x142, "/dev/pcanusb1" );
```

---

**10.22.4.121 void cSDH::OpenCAN\_PEEK (PCAN\_HANDLE \_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:**

- \_handle** : The PEAK CAN handle to reuse to connect to the PEAK CAN driver
- \_timeout** : The timeout to use:
  - <= 0 : wait forever (default)
  - T : wait for T seconds
- \_id\_read** - the CAN ID to use for reading (The SDH sends data on this ID, default=43=0x2a)
- \_id\_write** - the CAN ID to use for writing (The SDH receives data on this ID, default=42=0x2a)
- \_device** - the PEAK device name. Used for the Linux char dev driver only. default="/dev/pcanusb0"

**Examples:**

```
// Assuming 'hand' is a cSDH object ...
// and 'handle' is a valid PEAK NTCAN_HANDLE

// use default parameters for timeout and IDs
hand.OpenCAN_PEEK( handle );

// or use non default settings:
// timeout=1.0, id_read=0x143, id_write=0x142
hand.OpenCAN_PEEK( handle, 1.0, 0x143, 0x142 );
```

---

**10.22.4.122 void SDH::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:**

- leave\_enabled** - 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 );
```

**10.22.4.123 virtual bool SDH::cSDH::IsOpen (void) throw () [virtual]**

Return true if connection to SDH firmware/hardware is open.

Implements [SDH::cSDHBase](#).

**10.22.4.124 void SDH::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();
```

**10.22.4.125 void SDH::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();
```

#### 10.22.4.126 void SDH::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:**

*controller* - 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 );
```

**10.22.4.127 eControllerType SDH::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";
```

**10.22.4.128 void SDH::cSDH::SetVelocityProfile (eVelocityProfile *velocity\_profile*) throw (cSDHLibraryException\*)**

Set the type of velocity profile to be used in the SDH

**Parameters:**

*velocity\_profile* - 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 )
```

**10.22.4.129 eVelocityProfile SDH::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
```

---

**10.22.4.130 void SDH::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:**

*axes* - A vector of axis indices to access.

*motor\_currents* - 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](#).

*mode* - 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 );

```

#### 10.22.4.131 void SDH::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.

#### 10.22.4.132 std::vector<double> SDH::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:

*axes* - A vector of axis indices to access.

*mode* - 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};

```

#### 10.22.4.133 double SDH::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.

#### 10.22.4.134 void SDH::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 off when not needed.

##### Parameters:

*axes* - A vector of axis indices to access.

*states* - 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 );

```

#### 10.22.4.135 void SDH::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.

#### 10.22.4.136 void SDH::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.

#### 10.22.4.137 void SDH::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.

#### 10.22.4.138 std::vector<double> SDH::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:

*axes* - 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
```

**10.22.4.139 double SDH::cSDH::GetAxisEnable (int *iAxis*) throw (cSDHLibraryException\*)**

Like [GetAxisEnable\(std::vector<int>const&\)](#), just for a single axis *iAxis*, see there for details and examples.

**10.22.4.140 std::vector<eAxisState> SDH::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:**

*axes* - 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_IDLE, eAS_IDLE, eAS_DISABLE}

// 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_DISABLE}
```

#### **10.22.4.141 eAxisState SDH::cSDH::GetAxisActualState (int *iAxis*) throw (cSDHLibraryException\*)**

Like [GetAxisActualState\(std::vector<int>const&\)](#), just for a single axis *iAxis*, see there for details and examples.

#### **10.22.4.142 void SDH::cSDH::WaitAxis (std::vector< int > const & *axes*, double *timeout* = -1.0) throw (cSDHLibraryException\*)**

Wait until the movement(s) of of axis(axes) has finished

The state of the given axis(axes) is(are) queried until all axes are no longer moving.

#### Parameters:

*axes* - A vector of axis indices to access.  
*timeout* - 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 shure 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

// The previous command returned immediately, so
// before doing something else with the hand make shure the
// selected axes have stopped:
hand.WaitAxis( axes123 );

// now the axes have stopped

```

#### 10.22.4.143 void SDH::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 axes.

#### 10.22.4.144 void SDH::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:

*axes* - A vector of axis indices to access.

*angles* - 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 [0 .. *f\_max\_angle\_v*], i.e. it is checked that *angles[i]*, converted to internal units, is in [0 .. *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°:
hand.SetAxisTargetAngle( 3, -42.0 );

// Set target angle of for axis 0, 4 and 2 to 0.0°, -44.4° and -2.22° 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° (home-position)
hand.SetAxisTargetAngle( hand.All, 0.0 );
```

#### **10.22.4.145 void SDH::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.

#### 10.22.4.146 `std::vector<double> SDH::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:**

*axes* - 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}
```

#### 10.22.4.147 `double SDH::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.

#### 10.22.4.148 `std::vector<double> SDH::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:**

*axes* - 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}
```

**10.22.4.149 double SDH::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.

**10.22.4.150 void SDH::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. This means that in eCT\_VELOCITY\_ACCELERATION controller type the accelerations must be set with [SetAxisTargetAcceleration\(\)](#) before calling [SetAxisTargetVelocity\(\)](#).

**Parameters:**

*axes* - A vector of axis indices to access.

*velocities* - 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 [uc\\_angular\\_velocity](#).

**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 [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\[i\]\]](#)].
- 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°/s:
hand.SetAxisTargetVelocity( 3, 42.0 );

// Set target velocity of for axis 0,4 and 2 to 0.0°/s, 44.4°/s and 2.22°/s 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°/s
hand.SetAxisTargetVelocity( hand.All, 47.11 );
```

#### 10.22.4.151 void SDH::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.

#### 10.22.4.152 std::vector<double> SDH::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:

*axes* - 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}

// 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}
```

#### 10.22.4.153 double SDH::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.

#### 10.22.4.154 std::vector<double> SDH::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:

*axes* - 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, 120.0}

// 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}
```

#### 10.22.4.155 double SDH::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.

#### **10.22.4.156 std::vector<double> SDH::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:**

*axes* - 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, 120.0}

// 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}
```

#### **10.22.4.157 double SDH::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.

#### **10.22.4.158 std::vector<double> SDH::cSDH::GetAxisActualVelocity (std::vector< int >const & axes) throw (cSDHLibraryException\*)**

Get the actual velocity(s) of axis(axes).

**Parameters:**

*axes* - 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
```

#### 10.22.4.159 double SDH::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.

#### 10.22.4.160 std::vector<double> SDH::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:**

*axes* - 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.

#### **10.22.4.161 double SDH::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.

#### **10.22.4.162 void SDH::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:

*axes* - A vector of axis indices to access.

*accelerations* - 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°/s²:
hand.SetAxisTargetAcceleration( 3, 420.0 );

// Set target acceleration of for axis 0,4 and 2 to 0.0°/s², 444.0°/s² and 222°/s² 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°/s
hand.SetAxisTargetAcceleration( hand.All, 142.1 );

```

#### **10.22.4.163 void SDH::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.

#### **10.22.4.164 std::vector<double> SDH::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:**

*axes* - 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, 100.0}

// 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}
```

#### 10.22.4.165 double SDH::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.

#### 10.22.4.166 std::vector<double> SDH::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:

*axes* - 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}

// 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, -1.5707963267948966, -1.5707963267948966}

```

#### 10.22.4.167 double SDH::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.

#### 10.22.4.168 std::vector<double> SDH::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:

*axes* - 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}

// 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 }

```

**10.22.4.169 double SDH::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.

**10.22.4.170 std::vector<double> SDH::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:**

*axes* - 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}
```

#### 10.22.4.171 double SDH::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.

#### 10.22.4.172 std::vector<double> SDH::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:

*axes* - 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, 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}
```

#### 10.22.4.173 double SDH::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.

#### 10.22.4.174 double SDH::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:

*axes* - A vector of axis indices to access.

*sequ* - 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 addressing 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°. The
// target poses for axes 4 and 2 are still set since the
// last MoveAxes() call was sequentially (and 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 (and 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**

#### 10.22.4.175 double SDH::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).

#### 10.22.4.176 void SDH::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 off when not needed.

##### Parameters:

- fingers* - A vector of finger indices to access.
- states* - 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 );
```

---

**10.22.4.177 void SDH::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.

**10.22.4.178 void SDH::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.

**10.22.4.179 void SDH::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.

**10.22.4.180 std::vector<double> SDH::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:**

*fingers* - 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
```

**10.22.4.181 double SDH::cSDH::GetFingerEnable (int *iFinger*) throw (cSDHLibraryException\*)**

Like [GetFingerEnable\(std::vector<int>const&\)](#), just for a single finger *iFinger* and returning a single double value

**10.22.4.182 void SDH::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:**

*iFinger* - index of finger to access. This must be a single index.

*angles* - the angle(s) to set or *None* 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° }
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° }
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°
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!

```

#### **10.22.4.183 void SDH::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*.

#### **10.22.4.184 std::vector<double> SDH::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:**

*iFinger* - 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.

```

**10.22.4.185 void SDH::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.

**10.22.4.186 std::vector<double> SDH::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:**

*iFinger* - 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.
```

**10.22.4.187 void SDH::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.

**10.22.4.188 std::vector<double> SDH::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:**

*iFinger* - 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}
```

#### **10.22.4.189 void SDH::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.

#### **10.22.4.190 std::vector<double> SDH::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:**

*iFinger* - 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}
```

**10.22.4.191 void SDH::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.

**10.22.4.192 std::vector<double> SDH::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:**

- iFinger* - index of finger to access. This must be a single index  
*angles* - 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, 32.600000000000001, 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, 119.60000000000002, -53.0}

// Or if you change the angle unit system:
hand.UseRadians();
position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.5707963267948966, -1.5707963267948966
// now position is still something like {18.821618775581804, 119.60000000000002, -53.0}

// Or if you change the position unit system too:
hand.uc_position = &cSDH::uc_position_meter
position = hand.GetFingerXYZ( 0, 1.5707963267948966, -1.5707963267948966, -1.5707963267948966
// now position is still something like {0.018821618775581, 0.119.60000000000002, -0.05299999}
```

**10.22.4.193 std::vector<double> SDH::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*.

**10.22.4.194 double SDH::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:**

*fingers* - A vector of finger indices to access.  
*sequ* - 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 finger 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 overwritten
// 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**

#### **10.22.4.195 double SDH::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).

#### **10.22.4.196 double SDH::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**

#### **10.22.4.197 double SDH::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
```

**10.22.4.198 double SDH::cSDH::GripHand (eGraspId *grip*, double *close*, double *velocity*, bool *sequ* = 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 a 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:

*grip* - The index of the grip to perform [0..eGID\_DIMENSION-1] (s.a. [eGraspId](#))

*close* - close-ratio: [0.0 .. 1.0] where 0.0 is 'fully opened' and 1.0 is 'fully closed'

*velocity* - maximum allowed angular axis velocity in the chosen external unit system [uc\\_angular\\_velocity](#)

*sequ* - 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 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.
- 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 centrical grip at 50°/s:
hand.GripHand( hand.eGID_CENTRICAL, 0.0, 50.0, true );

// Now close it 50% with 30°/s:
hand.GripHand( hand.eGID_CENTRICAL, 0.5, 30.0, true );

// Then close it completely with 20°/s:
hand.GripHand( hand.eGID_CENTRICAL, 1.0, 20.0, true );
```

## 10.22.5 Member Data Documentation

### 10.22.5.1 static cUnitConverter const SDH::cSDH::uc\_angle\_degrees [static]

Default converter for angles (internal unit == external unit): degrees.

### 10.22.5.2 static cUnitConverter const SDH::cSDH::uc\_angle\_radians [static]

Converter for angles: external unit = radians.

### 10.22.5.3 static cUnitConverter const SDH::cSDH::uc\_time\_seconds [static]

Default converter for times (internal unit == external unit): seconds.

### 10.22.5.4 static cUnitConverter const SDH::cSDH::uc\_time\_milliseconds [static]

Converter for times: external unit = milliseconds.

### 10.22.5.5 static cUnitConverter const SDH::cSDH::uc\_temperature\_celsius [static]

Default converter for temperatures (internal unit == external unit): degrees celsius.

### 10.22.5.6 static cUnitConverter const SDH::cSDH::uc\_temperature\_fahrenheit [static]

Converter for temperatures: external unit = degrees fahrenheit.

### 10.22.5.7 static cUnitConverter const SDH::cSDH::uc\_angular\_velocity\_degrees\_per\_second [static]

Default converter for angular velocities (internal unit == external unit): degrees / second.

**10.22.5.8 static cUnitConverter const SDH::cSDH::uc\_angular\_velocity\_radians\_per\_second [static]**

Converter for angular velocities: external unit = radians/second.

**10.22.5.9 static cUnitConverter const SDH::cSDH::uc\_angular\_acceleration\_degrees\_per\_second\_squared [static]**

Default converter for angular accelerations (internal unit == external unit): degrees / second.

**10.22.5.10 static cUnitConverter const SDH::cSDH::uc\_angular\_acceleration\_radians\_per\_second\_squared [static]**

Converter for angular velocities: external unit = radians/second.

**10.22.5.11 static cUnitConverter const SDH::cSDH::uc\_motor\_current\_ampere [static]**

Default converter for motor current (internal unit == external unit): Ampere.

**10.22.5.12 static cUnitConverter const SDH::cSDH::uc\_motor\_current\_milliampere [static]**

Converter for motor current: external unit = milli Ampere.

**10.22.5.13 static cUnitConverter const SDH::cSDH::uc\_position\_millimeter [static]**

Default converter for position (internal unit == external unit): millimeter.

**10.22.5.14 static cUnitConverter const SDH::cSDH::uc\_position\_meter [static]**

Converter for position: external unit = meter.

**10.22.5.15 int SDH::cSDH::NUMBER\_OF\_AXES\_PER\_FINGER [protected]**

The number of axis per finger (for finger 1 this includes the "virtual" base axis).

**10.22.5.16 int SDH::cSDH::NUMBER\_OF\_VIRTUAL\_AXES [protected]**

The number of virtual axes.

**10.22.5.17 int SDH::cSDH::nb\_all\_axes [protected]**

The number of all axes including virtual axes.

**10.22.5.18 std::vector<int> SDH::cSDH::finger\_number\_of\_axes [protected]**

Mapping of finger index to number of real axes of fingers:.

**10.22.5.19 std::vector<std::vector<int> > SDH::cSDH::finger\_axis\_index [protected]**

Mapping of finger index, finger axis index to axis index::

**10.22.5.20 std::vector<double> SDH::cSDH::f\_zeros\_v [protected]**

Vector of 3 epsilon values.

Vector of 3 0.0 values

**10.22.5.21 std::vector<double> SDH::cSDH::f\_ones\_v [protected]**

Vector of 3 1.0 values.

**10.22.5.22 std::vector<double> SDH::cSDH::zeros\_v [protected]**

Vector of nb\_all\_axes 0.0 values.

**10.22.5.23 std::vector<double> SDH::cSDH::ones\_v [protected]**

Vector of nb\_all\_axes 1.0 values.

**10.22.5.24 std::vector<double> SDH::cSDH::f\_min\_motor\_current\_v [protected]**

Minimum allowed motor currents (in internal units (Ampere)), including the virtual axis.

**10.22.5.25 std::vector<double> SDH::cSDH::f\_max\_motor\_current\_v [protected]**

Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.

**10.22.5.26 std::vector<double> SDH::cSDH::f\_min\_angle\_v [protected]**

Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.

**10.22.5.27 std::vector<double> SDH::cSDH::f\_max\_angle\_v [protected]**

Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.

**10.22.5.28 std::vector<double> SDH::cSDH::f\_min\_velocity\_v [protected]**

Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

**10.22.5.29 std::vector<double> SDH::cSDH::f\_max\_velocity\_v [protected]**

Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

**10.22.5.30 std::vector<double> SDH::cSDH::f\_min\_acceleration\_v [protected]**

Minimum allowed axis acceleration (in internal units (degrees/(second \* second))), including the virtual axis.

**10.22.5.31 std::vector<double> SDH::cSDH::f\_max\_acceleration\_v [protected]**

Maximum allowed axis acceleration (in internal units (degrees/(second \* second))), including the virtual axis.

**10.22.5.32 double SDH::cSDH::grip\_max\_velocity [protected]**

Maximum allowed grip velocity (in internal units (degrees/second)).

**10.22.5.33 double SDH::cSDH::l1 [protected]**

length of limb 1 (proximal joint to distal joint) in mm

**10.22.5.34 double SDH::cSDH::l2 [protected]**

length of limb 2 (distal joint to fingertip) in mm

**10.22.5.35 double SDH::cSDH::d [protected]****10.22.5.36 double SDH::cSDH::h [protected]****10.22.5.37 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

**10.22.5.38 cSerialBase\* SDH::cSDH::com [protected]****10.22.5.39 cSDHSerial SDH::cSDH::comm\_interface**

The object to interface with the SDH attached via serial RS232 or CAN.

**10.22.5.40 std::vector<int> SDH::cSDH::all\_axes**

A vector with indices of all axes (in natural order), including the virtual axis.

**10.22.5.41 std::vector<int> SDH::cSDH::all\_real\_axes**

A vector with indices of all real axes (in natural order), excluding the virtual axis.

**10.22.5.42 std::vector<int> SDH::cSDH::all\_fingers**

A vector with indices of all fingers (in natural order).

**10.22.5.43 std::vector<int> SDH::cSDH::all\_temperature\_sensors**

A vector with indices of all temperature sensors.

**10.22.5.44 const cUnitConverter\* SDH::cSDH::uc\_angle**

unit convert for (axis) angles: default = [SDH::cSDH::uc\\_angle\\_degrees](#)

**10.22.5.45 const cUnitConverter\* SDH::cSDH::uc\_angular\_velocity**

unit convert for (axis) angular velocities: default = [SDH::cSDH::uc\\_angular\\_velocity\\_degrees\\_per\\_second](#)

**10.22.5.46 const cUnitConverter\* SDH::cSDH::uc\_angular\_acceleration**

unit convert for (axis) angular accelerations: default = [SDH::cSDH::uc\\_angular\\_acceleration\\_degrees\\_per\\_second\\_squared](#)

**10.22.5.47 const cUnitConverter\* SDH::cSDH::uc\_time**

unit convert for times: default = uc\_time\_seconds

**10.22.5.48 const cUnitConverter\* SDH::cSDH::uc\_temperature**

unit convert for temperatures: default = [SDH::cSDH::uc\\_temperature\\_celsius](#)

**10.22.5.49 const cUnitConverter\* SDH::cSDH::uc\_motor\_current**

unit converter for motor current: default = [SDH::cSDH::uc\\_motor\\_current\\_ampere](#)

**10.22.5.50 const cUnitConverter\* SDH::cSDH::uc\_position**

unit converter for position: default = [SDH::cSDH::uc\\_position\\_millimeter](#)

**10.22.5.51 std::vector<int> SDH::cSDH::finger\_number\_of\_axes [protected]**

Mapping of finger index to number of real axes of fingers:.

**10.22.5.52 std::vector<std::vector<int>> SDH::cSDH::finger\_axis\_index [protected]**

Mapping of finger index, finger axis index to axis index:.

**10.22.5.53 std::vector<double> SDH::cSDH::f\_zeros\_v [protected]**

Vector of 3 epsilon values.

Vector of 3 0.0 values

**10.22.5.54 std::vector<double> SDH::cSDH::f\_ones\_v [protected]**

Vector of 3 1.0 values.

**10.22.5.55 std::vector<double> SDH::cSDH::zeros\_v [protected]**

Vector of nb\_all\_axes 0.0 values.

**10.22.5.56 std::vector<double> SDH::cSDH::ones\_v [protected]**

Vector of nb\_all\_axes 1.0 values.

**10.22.5.57 std::vector<double> SDH::cSDH::f\_min\_motor\_current\_v [protected]**

Minimum allowed motor currents (in internal units (Ampere)), including the virtual axis.

**10.22.5.58 std::vector<double> SDH::cSDH::f\_max\_motor\_current\_v [protected]**

Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.

**10.22.5.59 std::vector<double> SDH::cSDH::f\_min\_angle\_v [protected]**

Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.

**10.22.5.60 std::vector<double> SDH::cSDH::f\_max\_angle\_v [protected]**

Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.

**10.22.5.61 std::vector<double> SDH::cSDH::f\_min\_velocity\_v [protected]**

Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

**10.22.5.62 std::vector<double> SDH::cSDH::f\_max\_velocity\_v [protected]**

Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis.

**10.22.5.63 std::vector<double> SDH::cSDH::f\_min\_acceleration\_v [protected]**

Minimum allowed axis acceleration (in internal units (degrees/(second \* second))), including the virtual axis.

**10.22.5.64 std::vector<double> SDH::cSDH::f\_max\_acceleration\_v [protected]**

Maximum allowed axis acceleration (in internal units (degrees/(second \* second))), including the virtual axis.

**10.22.5.65 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

**10.22.5.66 cSerialBase\* SDH::cSDH::com [protected]****10.22.5.67 std::vector<int> SDH::cSDH::all\_axes**

A vector with indices of all axes (in natural order), including the virtual axis.

**10.22.5.68 std::vector<int> SDH::cSDH::all\_real\_axes**

A vector with indices of all real axes (in natural order), excluding the virtual axis.

**10.22.5.69 std::vector<int> SDH::cSDH::all\_fingers**

A vector with indices of all fingers (in natural order).

**10.22.5.70 std::vector<int> SDH::cSDH::all\_temperature\_sensors**

A vector with indices of all temperature sensors.

**10.22.5.71 const cUnitConverter\* SDH::cSDH::uc\_angle**

unit convert for (axis) angles: default = [SDH::cSDH::uc\\_angle\\_degrees](#)

**10.22.5.72 const cUnitConverter\* SDH::cSDH::uc\_angular\_velocity**

unit convert for (axis) angular velocities: default = [SDH::cSDH::uc\\_angular\\_velocity\\_degrees\\_per\\_second](#)

**10.22.5.73 const cUnitConverter\* SDH::cSDH::uc\_angular\_acceleration**

unit convert for (axis) angular accelerations: default = [SDH::cSDH::uc\\_angular\\_acceleration\\_degrees\\_per\\_second\\_squared](#)

**10.22.5.74 const cUnitConverter\* SDH::cSDH::uc\_time**

unit convert for times: default = uc\_time\_seconds

**10.22.5.75 const cUnitConverter\* SDH::cSDH::uc\_temperature**

unit convert for temperatures: default = [SDH::cSDH::uc\\_temperature\\_celsius](#)

**10.22.5.76 const cUnitConverter\* SDH::cSDH::uc\_motor\_current**

unit converter for motor current: default = [SDH::cSDH::uc\\_motor\\_current\\_ampere](#)

**10.22.5.77 const cUnitConverter\* SDH::cSDH::uc\_position**

unit converter for position: default = [SDH::cSDH::uc\\_position\\_millimeter](#)

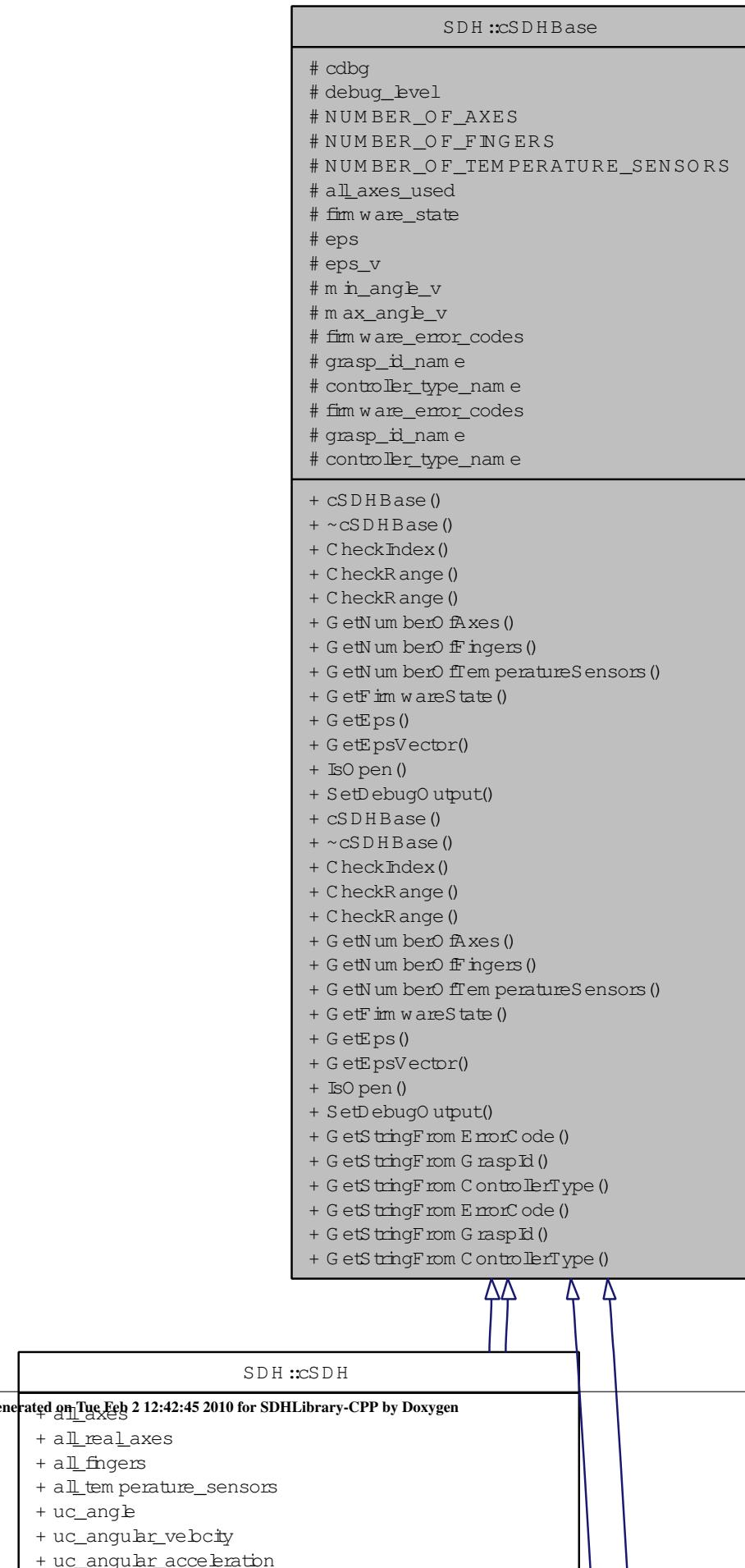
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[sdh.h](#)
- sdh/[sdh.h](#)
- cpp.desire.final/sdh/[sdh.cpp](#)
- sdh/[sdh.cpp](#)

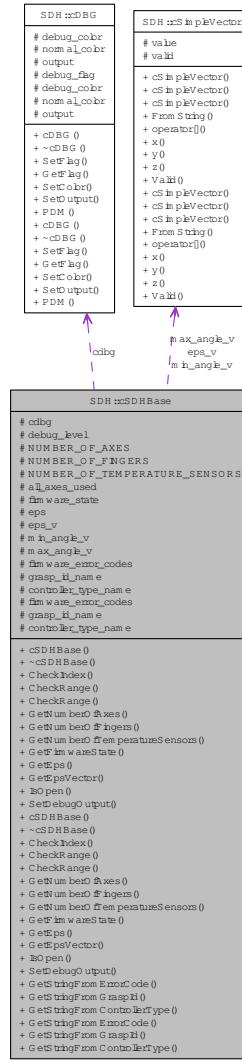
## 10.23 SDH::cSDHBase Class Reference

```
#include <sdhbase.h>
```

Inheritance diagram for SDH::cSDHBase:



Collaboration diagram for SDH::cSDHBase:



### 10.23.1 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.

#### Public Types

- enum { `All` = -1 }

*Anonymous enum (instead of define like macros).*

- enum `eErrorCode` {

```

eEC_SUCCESS = 0, eEC_NOT_AVAILABLE = 1, eEC_NO_SENSOR = 2, eEC_NOT_INITIALIZED = 3,
eEC_ALREADY_RUNNING = 4, eEC_FEATURE_NOT_SUPPORTED = 5, eEC_INCONSISTENT_DATA = 6, eEC_TIMEOUT = 7,
eEC_READ_ERROR = 8, eEC_WRITE_ERROR = 9, eEC_INSUFFICIENT_RESOURCES = 10,
eEC_CHECKSUM_ERROR = 11,
eEC_NOT_ENOUGH_PARAMS = 12, eEC_NO_PARAMS_EXPECTED = 13, eEC_CMD_UNKNOWN = 14, eEC_CMD_FORMAT_ERROR = 15,
eEC_ACCESS_DENIED = 16, eEC_ALREADY_OPEN = 17, eEC_CMD_FAILED = 18, eEC_CMD_ABORTED = 19,
eEC_INVALID_HANDLE = 20, eEC_DEVICE_NOT_FOUND = 21, eEC_DEVICE_NOT_OPENED = 22, eEC_IO_ERROR = 23,
eEC_INVALID_PARAMETER = 24, eEC_RANGE_ERROR = 25, eEC_NO_DATAPIPE = 26,
eEC_INDEX_OUT_OF_BOUNDS = 27,
eEC_HOMING_ERROR = 28, eEC_AXIS_DISABLED = 29, eEC_OVER_TEMPERATURE = 30,
eEC_DIMENSION,
eEC_SUCCESS = 0, eEC_NOT_AVAILABLE = 1, eEC_NO_SENSOR = 2, eEC_NOT_INITIALIZED = 3,
eEC_ALREADY_RUNNING = 4, eEC_FEATURE_NOT_SUPPORTED = 5, eEC_INCONSISTENT_DATA = 6, eEC_TIMEOUT = 7,
eEC_READ_ERROR = 8, eEC_WRITE_ERROR = 9, eEC_INSUFFICIENT_RESOURCES = 10,
eEC_CHECKSUM_ERROR = 11,
eEC_NOT_ENOUGH_PARAMS = 12, eEC_NO_PARAMS_EXPECTED = 13, eEC_CMD_UNKNOWN = 14, eEC_CMD_FORMAT_ERROR = 15,
eEC_ACCESS_DENIED = 16, eEC_ALREADY_OPEN = 17, eEC_CMD_FAILED = 18, eEC_CMD_ABORTED = 19,
eEC_INVALID_HANDLE = 20, eEC_DEVICE_NOT_FOUND = 21, eEC_DEVICE_NOT_OPENED = 22, eEC_IO_ERROR = 23,
eEC_INVALID_PARAMETER = 24, eEC_RANGE_ERROR = 25, eEC_NO_DATAPIPE = 26,
eEC_INDEX_OUT_OF_BOUNDS = 27,
eEC_HOMING_ERROR = 28, eEC_AXIS_DISABLED = 29, eEC_OVER_TEMPERATURE = 30,
eEC_DIMENSION }

• enum eGraspId {
    eGID_INVALID = -1, eGID_CENTRICAL = 0, eGID_PARALLEL = 1, eGID_CYLINDRICAL = 2,
    eGID_SPHERICAL = 3, eGID_DIMENSION, 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, 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,`  
`eVP_INVALID = -1, eVP_SIN_SQUARE, eVP_RAMP, eVP_DIMENSION }`

*An enum for all possible SDH internal velocity profile types.*
- enum { `All = -1` }
 

*Anonymous enum (instead of define like macros).*
- enum `eErrorCode` {
   
`eEC_SUCCESS = 0, eEC_NOT_AVAILABLE = 1, eEC_NO_SENSOR = 2, eEC_NOT_INITIALIZED = 3,`  
`eEC_ALREADY_RUNNING = 4, eEC_FEATURE_NOT_SUPPORTED = 5, eEC_INCONSISTENT_DATA = 6, eEC_TIMEOUT = 7,`  
`eEC_READ_ERROR = 8, eEC_WRITE_ERROR = 9, eEC_INSUFFICIENT_RESOURCES = 10,`  
`eEC_CHECKSUM_ERROR = 11,`  
`eEC_NOT_ENOUGH_PARAMS = 12, eEC_NO_PARAMS_EXPECTED = 13, eEC_CMD_UNKNOWN = 14, eEC_CMD_FORMAT_ERROR = 15,`  
`eEC_ACCESS_DENIED = 16, eEC_ALREADY_OPEN = 17, eEC_CMD_FAILED = 18, eEC_CMD_ABORTED = 19,`  
`eEC_INVALID_HANDLE = 20, eEC_DEVICE_NOT_FOUND = 21, eEC_DEVICE_NOT_OPENED = 22, eEC_IO_ERROR = 23,`  
`eEC_INVALID_PARAMETER = 24, eEC_RANGE_ERROR = 25, eEC_NO_DATAPIPE = 26,`  
`eEC_INDEX_OUT_OF_BOUNDS = 27,`  
`eEC_HOMING_ERROR = 28, eEC_AXIS_DISABLED = 29, eEC_OVER_TEMPERATURE = 30,`  
`eEC_DIMENSION,`  
`eEC_SUCCESS = 0, eEC_NOT_AVAILABLE = 1, eEC_NO_SENSOR = 2, eEC_NOT_INITIALIZED = 3,`  
`eEC_ALREADY_RUNNING = 4, eEC_FEATURE_NOT_SUPPORTED = 5, eEC_INCONSISTENT_DATA = 6, eEC_TIMEOUT = 7,`  
`eEC_READ_ERROR = 8, eEC_WRITE_ERROR = 9, eEC_INSUFFICIENT_RESOURCES = 10,`  
`eEC_CHECKSUM_ERROR = 11,`  
`eEC_NOT_ENOUGH_PARAMS = 12, eEC_NO_PARAMS_EXPECTED = 13, eEC_CMD_UNKNOWN = 14, eEC_CMD_FORMAT_ERROR = 15,`  
`eEC_ACCESS_DENIED = 16, eEC_ALREADY_OPEN = 17, eEC_CMD_FAILED = 18, eEC_CMD_ABORTED = 19,`  
`eEC_INVALID_HANDLE = 20, eEC_DEVICE_NOT_FOUND = 21, eEC_DEVICE_NOT_OPENED = 22, eEC_IO_ERROR = 23,`  
`eEC_INVALID_PARAMETER = 24, eEC_RANGE_ERROR = 25, eEC_NO_DATAPIPE = 26,`  
`eEC_INDEX_OUT_OF_BOUNDS = 27,`  
`eEC_HOMING_ERROR = 28, eEC_AXIS_DISABLED = 29, eEC_OVER_TEMPERATURE = 30,`  
`eEC_DIMENSION }`
- enum `eGraspId` {
   
`eGID_INVALID = -1, eGID_CENTRICAL = 0, eGID_PARALLEL = 1, eGID_CYLINDRICAL = 2,`  
`eGID_SPHERICAL = 3, eGID_DIMENSION, 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, 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,`
  
`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*

- **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.*
- 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.*
- 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.*

## 10.23.2 Member Enumeration Documentation

### 10.23.2.1 anonymous enum

Anonymous enum (instead of define like macros).

#### Enumerator:

`All` A meta-value that means "access all possible values".

### 10.23.2.2 enum SDH::cSDHBase::eErrorCode

The error codes of the SDH firmware

#### Enumerator:

`eEC_SUCCESS`  
`eEC_NOT_AVAILABLE`

*eEC\_NO\_SENSOR*  
*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\_DIMENSION* Endmarker and dimension.  
*eEC\_SUCCESS*  
*eEC\_NOT\_AVAILABLE*  
*eEC\_NO\_SENSOR*  
*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\_DIMENSION* Endmarker and dimension.

#### 10.23.2.3 enum SDH::cSDHBase::eGraspId

The enum values of the known grasps.

**Enumerator:**

*eGID\_INVALID* invalid grasp id  
*eGID\_CENTRICAL* centrical grasp: ???  
*eGID\_PARALLEL* parallel grasp: ???  
*eGID\_CYLINDRICAL* cylindrical grasp: ???  
*eGID\_SPHERICAL* spherecial grasp: ???  
*eGID\_DIMENSION* Endmarker and dimension.  
*eGID\_INVALID* invalid grasp id  
*eGID\_CENTRICAL* centrical grasp: ???  
*eGID\_PARALLEL* parallel grasp: ???  
*eGID\_CYLINDRICAL* cylindrical grasp: ???  
*eGID\_SPHERICAL* spherecial grasp: ???  
*eGID\_DIMENSION* Endmarker and dimension.

#### 10.23.2.4 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.

**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.

#### 10.23.2.5 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

**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

### 10.23.2.6 anonymous enum

Anonymous enum (instead of define like macros).

Enumerator:

*All* A meta-value that means "access all possible values".

### 10.23.2.7 enum SDH::cSDHBase::eErrorCode

The error codes of the SDH firmware

Enumerator:

*eEC\_SUCCESS*  
*eEC\_NOT\_AVAILABLE*  
*eEC\_NO\_SENSOR*  
*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\_DIMENSION* Endmarker and dimension.  
*eEC\_SUCCESS*  
*eEC\_NOT\_AVAILABLE*  
*eEC\_NO\_SENSOR*  
*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\_DIMENSION* Endmarker and dimension.

#### 10.23.2.8 enum SDH::cSDHBase::eGraspId

The enum values of the known grasps.

Enumerator:

*eGID\_INVALID* invalid grasp id

*eGID\_CENTRICAL* centrical grasp: ???  
*eGID\_PARALLEL* parallel grasp: ???  
*eGID\_CYLINDRICAL* cylindrical grasp: ???  
*eGID\_SPHERICAL* spherecial grasp: ???  
*eGID\_DIMENSION* Endmarker and dimension.  
*eGID\_INVALID* invalid grasp id  
*eGID\_CENTRICAL* centrical grasp: ???  
*eGID\_PARALLEL* parallel grasp: ???  
*eGID\_CYLINDRICAL* cylindrical grasp: ???  
*eGID\_SPHERICAL* spherecial grasp: ???  
*eGID\_DIMENSION* Endmarker and dimension.

#### 10.23.2.9 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.  
*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.

#### 10.23.2.10 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
- 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

**10.23.3 Constructor & Destructor Documentation****10.23.3.1 cSDHBase::cSDHBase (int *debug\_level*)**

Constructor of [cSDHBase](#) class, initialize internal variables and settings

**Parameters:**

- debug\_level* : debug level of the created object. If the *debug\_level* of an object is  $> 0$  then it will output debug messages.
  - (Subclasses of [cSDHBase](#) like [cSDH](#) or [cSDHSerial](#) use additional settings, see there.)

**10.23.3.2 virtual SDH::cSDHBase::~cSDHBase () [inline, virtual]**

virtual destructor to make compiler happy

**10.23.3.3 SDH::cSDHBase::cSDHBase (int *debug\_level*)**

Constructor of [cSDHBase](#) class, initialize internal variables and settings

**Parameters:**

- debug\_level* : debug level of the created object. If the *debug\_level* of an object is  $> 0$  then it will output debug messages.
  - (Subclasses of [cSDHBase](#) like [cSDH](#) or [cSDHSerial](#) use additional settings, see there.)

**10.23.3.4 virtual SDH::cSDHBase::~cSDHBase () [inline, virtual]**

virtual destructor to make compiler happy

**10.23.4 Member Function Documentation****10.23.4.1 void cSDHBase::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.

**10.23.4.2 void cSDHBase::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.

**10.23.4.3 void cSDHBase::CheckRange (double \* *values*, double \* *minvalues*, double \* *maxvalues*, char const \* *name* = " ") throw (cSDHErrorInvalidParameter\*)**

Check if any value[i] in array *values* is in [*minvalue*[i] .. *maxvalue*[i]]. Throw a [cSDHErrorInvalidParameter](#) exception if not.

**10.23.4.4 char const \* cSDHBase::GetStringFromErrorCode (eErrorCode *error\_code*) [static]**

Return a ptr to a (static) string describing error code *error\_code*.

**10.23.4.5 char const \* cSDHBase::GetStringFromGraspId (eGraspId *grasp\_id*) [static]**

Return a ptr to a (static) string describing grasp id *grasp\_id*.

**10.23.4.6 char const \* cSDHBase::GetStringFromControllerType (eControllerType *controller\_type*) [static]**

Return a ptr to a (static) string describing controller type *controller\_Type*.

**10.23.4.7 int cSDHBase::GetNumberOfAxes (void)**

Return the number of axes of the SDH.

**10.23.4.8 int cSDHBase::GetNumberOfFingers (void)**

Return the number of fingers of the SDH.

**10.23.4.9 int cSDHBase::GetNumberOfTemperatureSensors (void)**

Return the number of temperature sensors of the SDH.

**10.23.4.10 cSDHBase::eErrorCode cSDHBase::GetFirmwareState (void)**

Return the last known state of the SDH firmware.

**10.23.4.11 double cSDHBase::GetEps (void)**

Return the [eps](#) value.

**10.23.4.12 cSimpleVector const & cSDHBase::GetEpsVector (void)**

Return simple vector of number of axes epsilon values.

**10.23.4.13 virtual bool SDH::cSDHBase::IsOpen (void) [pure virtual]**

Return true if connection to SDH firmware/hardware is open.

Implemented in [SDH::cSDH](#), [SDH::cSDHSerial](#), [SDH::cSDH](#), and [SDH::cSDHSerial](#).

**10.23.4.14 virtual void SDH::cSDHBase::SetDebugOutput (std::ostream \* *debuglog*) [inline, virtual]**

change the stream to use for debug messages

Reimplemented in [SDH::cSDH](#), and [SDH::cSDH](#).

**10.23.4.15 void SDH::cSDHBase::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.

**10.23.4.16 void SDH::cSDHBase::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.

**10.23.4.17 void SDH::cSDHBase::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.

**10.23.4.18 static char const\* SDH::cSDHBase::GetStringFromErrorCode (eErrorCode *error\_code*) [static]**

Return a ptr to a (static) string describing error code *error\_code*.

**10.23.4.19 static char const\* SDH::cSDHBase::GetStringFromGraspId (eGraspId *grasp\_id*) [static]**

Return a ptr to a (static) string describing grasp id *grasp\_id*.

**10.23.4.20 static char const\* SDH::cSDHBase::GetStringFromControllerType (eControllerType *controller\_type*) [static]**

Return a ptr to a (static) string describing controller type *controller\_Type*.

**10.23.4.21 int SDH::cSDHBase::GetNumberOfAxes (void)**

Return the number of axes of the SDH.

**10.23.4.22 int SDH::cSDHBase::GetNumberOfFingers (void)**

Return the number of fingers of the SDH.

**10.23.4.23 int SDH::cSDHBase::GetNumberOfTemperatureSensors (void)**

Return the number of temperature sensors of the SDH.

**10.23.4.24 eErrorCode SDH::cSDHBase::GetFirmwareState (void)**

Return the last known state of the SDH firmware.

**10.23.4.25 double SDH::cSDHBase::GetEps (void)**

Return the `eps` value.

**10.23.4.26 cSimpleVector const& SDH::cSDHBase::GetEpsVector (void)**

Return simple vector of number of axes epsilon values.

**10.23.4.27 virtual bool SDH::cSDHBase::IsOpen (void) [pure virtual]**

Return true if connection to SDH firmware/hardware is open.

Implemented in [SDH::cSDH](#), [SDH::cSDHSerial](#), [SDH::cSDH](#), and [SDH::cSDHSerial](#).

**10.23.4.28 virtual void SDH::cSDHBase::SetDebugOutput (std::ostream \* *debuglog*) [inline, virtual]**

change the stream to use for debug messages

Reimplemented in [SDH::cSDH](#), and [SDH::cSDH](#).

## 10.23.5 Member Data Documentation

**10.23.5.1 cDBG SDH::cSDHBase::cdbg [protected]**

debug stream to print colored debug messages

**10.23.5.2 int SDH::cSDHBase::debug\_level [protected]**

debug level of this object

**10.23.5.3 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_FEATUREeEC_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_DIMENSION: Number of error codes"
}
```

A mapping from [eErrorCode](#) error code enums to strings with human readable error messages.

**10.23.5.4 char const \* cSDHBase::grasp\_id\_name [static, protected]****Initial value:**

```
{
    "eGID_CENTRICAL: centrical 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.

**10.23.5.5 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.

#### **10.23.5.6 int SDH::cSDHBase::NUMBER\_OF\_AXES [protected]**

The number of axes.

#### **10.23.5.7 int SDH::cSDHBase::NUMBER\_OF\_FINGERS [protected]**

The number of fingers.

#### **10.23.5.8 int SDH::cSDHBase::NUMBER\_OF\_TEMPERATURE\_SENSORS [protected]**

The number of temperature sensors.

#### **10.23.5.9 int SDH::cSDHBase::all\_axes\_used [protected]**

Bit field with the bits for all axes set.

#### **10.23.5.10 eErrorCode SDH::cSDHBase::firmware\_state [protected]**

the last known state of the SDH firmware

#### **10.23.5.11 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)

#### **10.23.5.12 cSimpleVector SDH::cSDHBase::eps\_v [protected]**

simple vector of 7 epsilon values

#### **10.23.5.13 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))

**10.23.5.14 cSimpleVector SDH::cSDHBase::max\_angle\_v [protected]**

Maximum allowed axis angles (in internal units (degrees)).

**10.23.5.15 char const\* SDH::cSDHBase::firmware\_error\_codes[] [static, protected]**

A mapping from [eErrorCode](#) error code enums to strings with human readable error messages.

**10.23.5.16 char const\* SDH::cSDHBase::grasp\_id\_name[] [static, protected]**

A mapping from [eGraspId](#) grasp id enums to strings with human readable grasp id names.

**10.23.5.17 char const\* SDH::cSDHBase::controller\_type\_name[] [static, protected]**

A mapping from [eControllerType](#) controller type enums to strings with human readable controller type names.

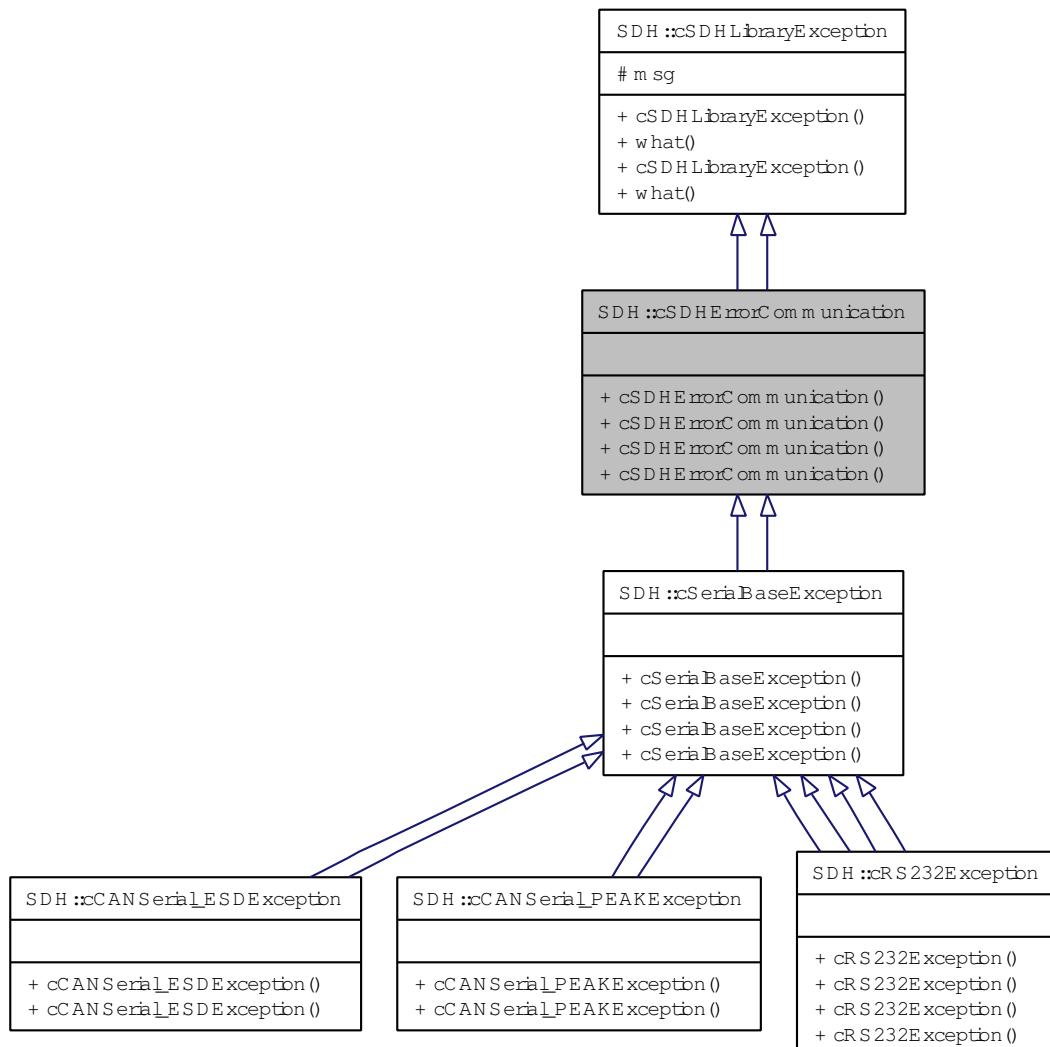
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[sdhbase.h](#)
- sdh/[sdhbase.h](#)
- cpp.desire.final/sdh/[sdhbase.cpp](#)
- sdh/[sdhbase.cpp](#)

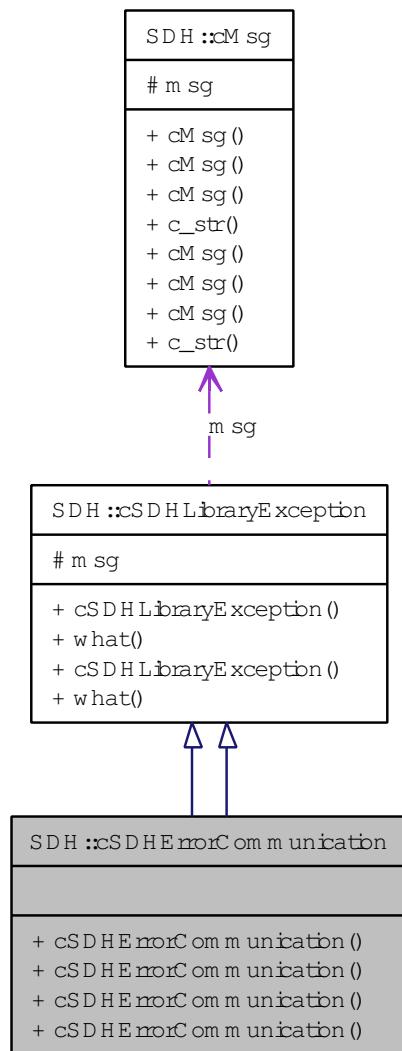
## 10.24 SDH::cSDHErrorCommunication Class Reference

```
#include <sdhexception.h>
```

Inheritance diagram for SDH::cSDHErrorCommunication:



Collaboration diagram for SDH::cSDHErrorCommunication:



### 10.24.1 Detailed Description

Derived exception class for exceptions related to communication between the SDHLibrary and the SDH.

#### Public Member Functions

- [cSDHErrorCommunication \(cMsg const &\\_msg\)](#)
- [cSDHErrorCommunication \(char const \\*\\_type, cMsg const &\\_msg\)](#)
- [cSDHErrorCommunication \(cMsg const &\\_msg\)](#)
- [cSDHErrorCommunication \(char const \\*\\_type, cMsg const &\\_msg\)](#)

## 10.24.2 Constructor & Destructor Documentation

**10.24.2.1 SDH::cSDHErrorCommunication::cSDHErrorCommunication (cMsg const & \_msg)**  
[inline]

**10.24.2.2 SDH::cSDHErrorCommunication::cSDHErrorCommunication (char const \* \_type,**  
**cMsg const & \_msg) [inline]**

**10.24.2.3 SDH::cSDHErrorCommunication::cSDHErrorCommunication (cMsg const & \_msg)**  
[inline]

**10.24.2.4 SDH::cSDHErrorCommunication::cSDHErrorCommunication (char const \* \_type,**  
**cMsg const & \_msg) [inline]**

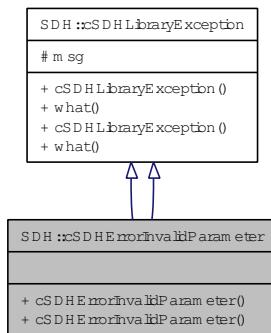
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[sdhexception.h](#)
- sdh/[sdhexception.h](#)

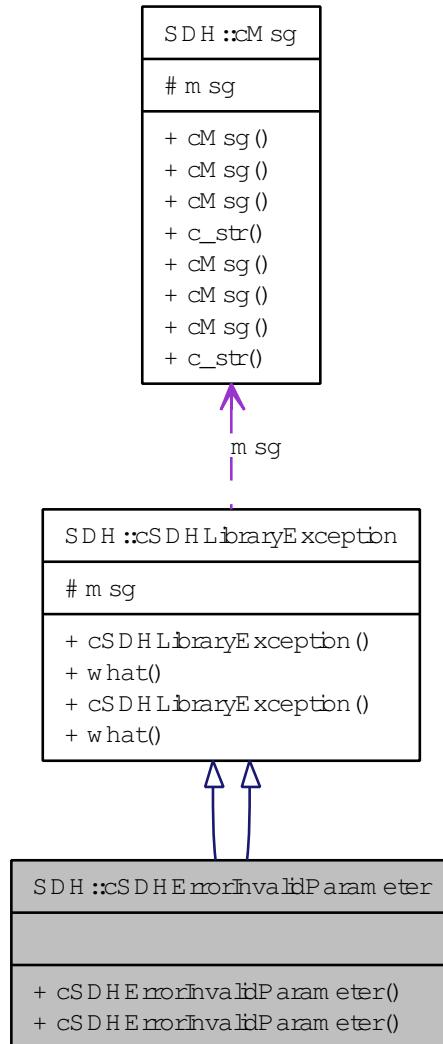
## 10.25 SDH::cSDHErrorInvalidParameter Class Reference

```
#include <sdhbase.h>
```

Inheritance diagram for SDH::cSDHErrorInvalidParameter:



Collaboration diagram for SDH::cSDHErrorInvalidParameter:



### 10.25.1 Detailed Description

Derived exception class for exceptions related to invalid parameters.

### Public Member Functions

- [cSDHErrorInvalidParameter \(cMsg const &\\_msg\)](#)
- [cSDHErrorInvalidParameter \(cMsg const &\\_msg\)](#)

## 10.25.2 Constructor & Destructor Documentation

**10.25.2.1 SDH::cSDHErrorInvalidParameter::cSDHErrorInvalidParameter (cMsg const & \_msg)**  
[inline]

**10.25.2.2 SDH::cSDHErrorInvalidParameter::cSDHErrorInvalidParameter (cMsg const & \_msg)**  
[inline]

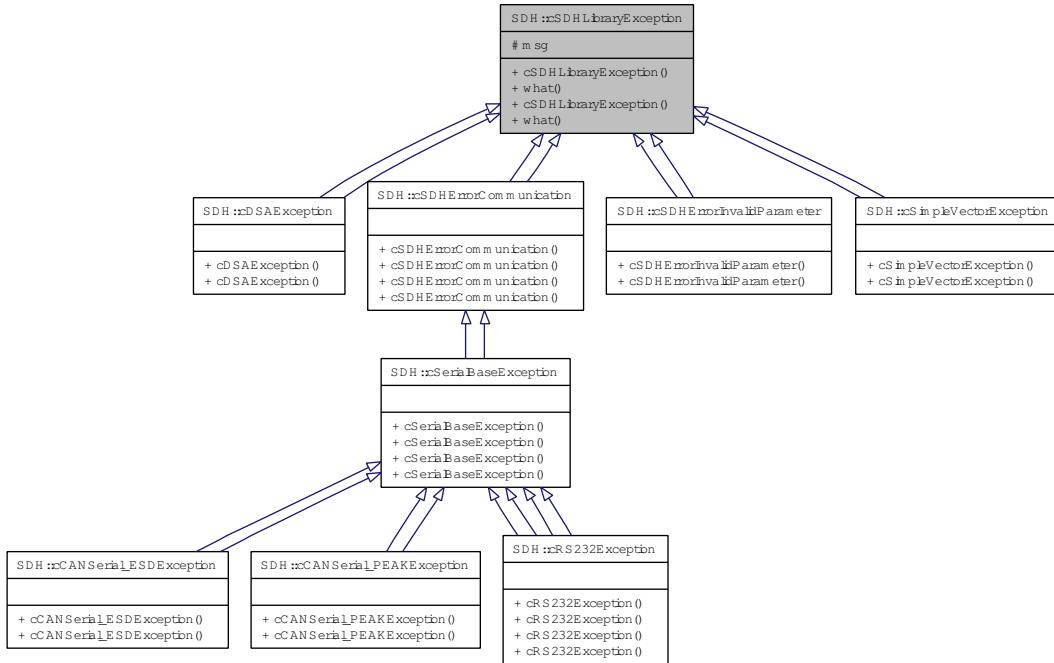
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[sdhbase.h](#)
- sdh/[sdhbase.h](#)

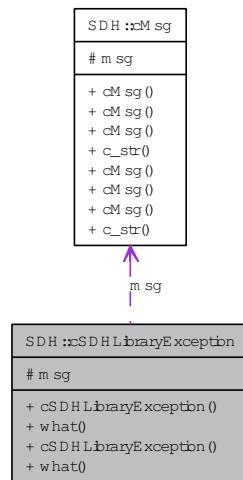
## 10.26 SDH::cSDHLibraryException Class Reference

```
#include <sdhexception.h>
```

Inheritance diagram for SDH::cSDHLibraryException:



Collaboration diagram for SDH::cSDHLibraryException:



### 10.26.1 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.

## Public Member Functions

- `cSDHLibraryException (char const *_type, cMsg const &_msg)`
- `virtual const char * what () const throw ()`
- `cSDHLibraryException (char const *_type, cMsg const &_msg)`
- `virtual const char * what () const throw ()`

## Protected Attributes

- `cMsg msg`  
*The message object.*

### 10.26.2 Constructor & Destructor Documentation

#### 10.26.2.1 `cSDHLibraryException::cSDHLibraryException (char const *_type, cMsg const &_msg)`

Constructor of sdh exception base class.

##### Parameters:

- `_type` - the type name of the exception. By convention this is the class name of the exception
- `_msg` - a reference to a `cMsg` object that further describes the exception.

##### Remarks:

- The `_type` parameter is mainly useful 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)" ) );
```

- 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 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;
}

```

### 10.26.2.2 SDH::cSDHLibraryException::cSDHLibraryException (char const \* *\_type*, cMsg const & *\_msg*)

Constructor of sdh exception base class.

#### Parameters:

- *\_type* - the type name of the exception. By convention this is the class name of the exception
- *\_msg* - 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 > %

```

- 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 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;
    }
```

### 10.26.3 Member Function Documentation

#### 10.26.3.1 const char \* SDH::cSDHLibraryException::what () const throw () [virtual]

Return the [msg](#) member

#### 10.26.3.2 virtual const char\* SDH::cSDHLibraryException::what () const throw () [virtual]

Return the [msg](#) member

### 10.26.4 Member Data Documentation

#### 10.26.4.1 cMsg SDH::cSDHLibraryException::msg [protected]

The message object.

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

- [cpp.desire.final/sdh/sdhexception.h](#)
- [sdh/sdhexception.h](#)
- [cpp.desire.final/sdh/sdhexception.cpp](#)
- [sdh/sdhexception.cpp](#)

## 10.27 cSDHOptions Class Reference

```
#include <sdhoptions.h>
```

### Public Member Functions

- `cSDHOptions` (char const \*option\_selection="general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon")
- int `Parse` (int argc, char \*\*argv, char const \*helptext, char const \*progname, char const \*version, char const \*libname, char const \*ibrelease)
- `cSDHOptions` (char const \*option\_selection="general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon")
- int `Parse` (int argc, char \*\*argv, char const \*helptext, char const \*progname, char const \*version, char const \*libname, char const \*ibrelease)

### 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`
- std::ostream \* `debuglog`

### Static Public Attributes

- static int const `MAX_DEV_LENGTH` = 32

## 10.27.1 Constructor & Destructor Documentation

**10.27.1.1 cSDHOptions::cSDHOptions (char const \* *option\_selection* =  
 "general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_-  
 peakcan sdhcom\_cancommon")**

constructor: init members to their default values

**Parameters:**

*option\_selection* - 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

- "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
- "sdhother" see sdhusage\_sdhother
- "dsacom" see sdhusage\_dsacom
- "dsaother" see sdhusage\_dsaother

**10.27.1.2 cSDHOptions::cSDHOptions (char const \* *option\_selection* =  
 "general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_-  
 peakcan sdhcom\_cancommon")**

constructor: init members to their default values

**Parameters:**

*option\_selection* - 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

- "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
- "sdhother" see sdhusage\_sdhother
- "dsacom" see sdhusage\_dsacom
- "dsaother" see sdhusage\_dsaother

## 10.27.2 Member Function Documentation

**10.27.2.1 int cSDHOptions::Parse (int *argc*, char \*\* *argv*, char const \* *helptext*, char const \*  
*progname*, char const \* *version*, char const \* *libname*, char const \* *librelease*)**

parse the command line parameters *argc*, *argv* into members. *helptext*, *progname*, *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

**10.27.2.2 int cSDHOptions::Parse (int argc, char \*\*argv, char const \*helptext, char const \*progname, char const \*version, char const \*libname, char const \*librelease)**

parse the command line parameters *argc*, *argv* into members. *helptext*, *progname*, *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



### 10.27.3 Member Data Documentation

**10.27.3.1 static int const cSDHOptions::MAX\_DEV\_LENGTH = 32 [static]**

**10.27.3.2 std::string cSDHOptions::usage**

**10.27.3.3 int cSDHOptions::debug\_level**

**10.27.3.4 std::ostream\* cSDHOptions::debuglog**

**10.27.3.5 int cSDHOptions::sdhport**

**10.27.3.6 char cSDHOptions::sdh\_rs\_device**

**10.27.3.7 double cSDHOptions::timeout**

**10.27.3.8 unsigned long cSDHOptions::rs232\_baudrate**

**10.27.3.9 bool cSDHOptions::use\_can\_esd**

**10.27.3.10 int cSDHOptions::net**

**10.27.3.11 bool cSDHOptions::use\_can\_peak**

**10.27.3.12 char cSDHOptions::sdh\_canpeak\_device**

**10.27.3.13 unsigned long cSDHOptions::can\_baudrate**

**10.27.3.14 unsigned int cSDHOptions::id\_read**

**10.27.3.15 unsigned int cSDHOptions::id\_write**

**10.27.3.16 bool cSDHOptions::use\_radians**

**10.27.3.17 bool cSDHOptions::use\_fahrenheit**

**10.27.3.18 double cSDHOptions::period**

**10.27.3.19 int cSDHOptions::dsaport**

**10.27.3.20 char cSDHOptions::dsa\_rs\_device**

**10.27.3.21 bool cSDHOptions::do\_RLE**

**10.27.3.22 int cSDHOptions::framerate**

**10.27.3.23 bool cSDHOptions::fullframe**

**10.27.3.24 bool cSDHOptions::sensorinfo**

**10.27.3.25 bool cSDHOptions::controllerinfo**

**10.27.3.26 int cSDHOptions::matrixinfo**

**10.27.3.27 std::ostream\* cSDHOptions::debuglog**

Generated on Tue Feb 2 12:42:45 2010 for SDHLibrary-CPP by Doxygen

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

- [cpp.desire.final/demo/sdhoptions.h](#)
- [demo/sdhoptions.h](#)
- [cpp.desire.final/demo/sdhoptions.cpp](#)
- [demo/sdhoptions.cpp](#)

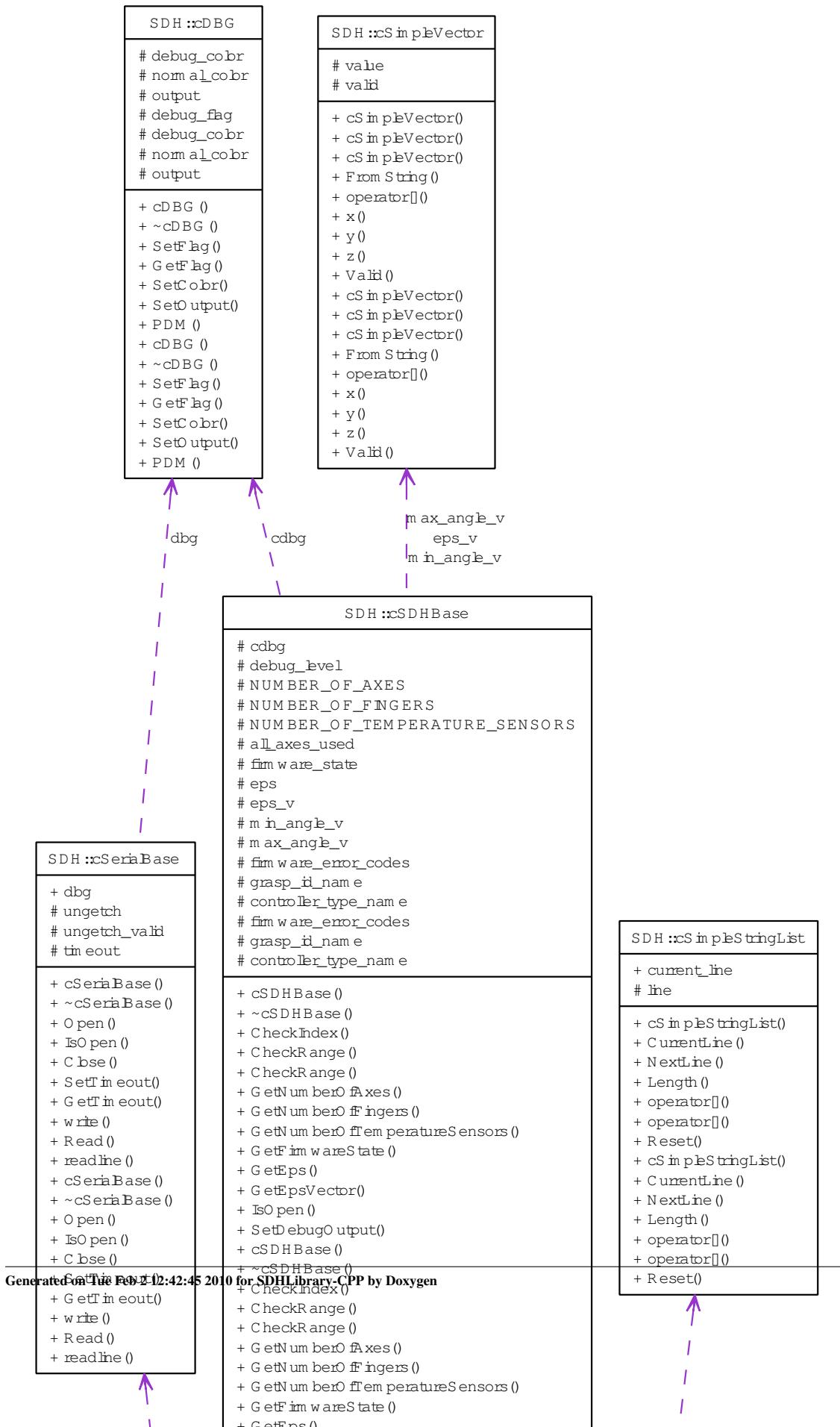
## 10.28 SDH::cSDHSerial Class Reference

```
#include <sdhserial.h>
```

## Inheritance diagram for SDH::cSDHSerial:



Collaboration diagram for SDH::cSDHSerial:



### 10.28.1 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.

## 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 SyncUnknown () throw (cSDHErrorCommunication*)`
- `cSimpleVector AxisCommand (char const *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 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*)`
- `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\*)

### 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 SyncUnknown ()` throw (cSDHErrorCommunication\*)
- `cSimpleVector AxisCommand (char const *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 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*`)
- 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*
- 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).*

## 10.28.2 Constructor & Destructor Documentation

### 10.28.2.1 **cSDHSerial::cSDHSerial (int \_debug\_level = 0)**

Constructor of [cSDHSerial](#).

**Parameters:**

*\_debug\_level* : debug level of the created object. If the *debug\_level* 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

### 10.28.2.2 **virtual SDH::cSDHSerial::~cSDHSerial () [inline, virtual]**

virtual destructor to make compiler happy

### 10.28.2.3 **SDH::cSDHSerial::cSDHSerial (int \_debug\_level = 0)**

Constructor of [cSDHSerial](#).

**Parameters:**

*\_debug\_level* : debug level of the created object. If the *debug\_level* of an object is > 0 then it will output debug messages. (forwarded to constructor of base class)

### 10.28.2.4 **virtual SDH::cSDHSerial::~cSDHSerial () [inline, virtual]**

virtual destructor to make compiler happy

## 10.28.3 Member Function Documentation

### 10.28.3.1 **void cSDHSerial::Open (cSerialBase \* \_com) throw (cSDHLibraryException\*)**

Open the serial device and check connection to SDH by querying the SDH firmware version

**Parameters:**

*\_com* - 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.

#### **10.28.3.2 void cSDHSerial::Close (void) throw (cSDHLibraryException\*)**

Close connection to serial port.

#### **10.28.3.3 bool cSDHSerial::IsOpen (void) [virtual]**

Return true if connection to SDH firmware/hardware is open

Implements [SDH::cSDHBase](#).

#### **10.28.3.4 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.

#### **10.28.3.5 void cSDHSerial::ExtractFirmwareState () throw ( cSDHErrorCommunication\* )**

Try to extract the state of the SDH firmware from the last reply

#### **10.28.3.6 double cSDHSerial::GetDuration (char \* line) throw ( cSDHErrorCommunication\* )**

Return duration of the execution of a SDH command as reported by line

#### **10.28.3.7 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

**10.28.3.8 void cSDHSerial::Sync () throw ( cSDHErrorCommunication\* )**

Read all pending lines from SDH to resync execution of PC and SDH.

**10.28.3.9 void cSDHSerial::SyncUnknown () throw ( cSDHErrorCommunication\* )**

Read an unknown number of lines from SDH to resync execution of PC and SDH.

**10.28.3.10 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.

**10.28.3.11 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).

**10.28.3.12 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.

---

**10.28.3.13 cSimpleVector cSDHSerial::ilim (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.

---

**10.28.3.14 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 single number or a boolean value then all axes power states are set to that value, the NUMBER\_OF\_AXES-list is returned.

---

**10.28.3.15 void cSDHSerial::demo (bool *onoff*)**

Enable/disable SCHUNK demo

---

**10.28.3.16 int cSDHSerial::property (char const \* *propname*, int *value*)**

Set named property

Valid propnames are:

- "user\_errors"
- "terminal"
- "debug"

**10.28.3.17 int cSDHSerial::user\_errors (int *value*)**

**10.28.3.18 int cSDHSerial::terminal (int *value*)**

**10.28.3.19 int cSDHSerial::debug (int *value*)**

**10.28.3.20 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**

**10.28.3.21 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.

**10.28.3.22 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).

#### **10.28.3.23 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.

#### **10.28.3.24 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.

#### **10.28.3.25 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

#### **10.28.3.26 void cSDHSerial::stop (void) throw (cSDHLibraryException\*)**

Stop sdh.

Will NOT interrupt a previous "selgrip" or "grip" command, only an "m" command!

**10.28.3.27 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.

**10.28.3.28 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.

**10.28.3.29 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.

**10.28.3.30 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

---

**10.28.3.31 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

**10.28.3.32 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.

**10.28.3.33 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.

**10.28.3.34 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.

**10.28.3.35 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.

**10.28.3.36 cSimpleVector cSDHSerial::temp\_electronics (void) throw (cSDHLibraryException\*)**

Get actual temperatures of the electronics, i.e. teh 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.

**10.28.3.37 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

**10.28.3.38 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

**10.28.3.39 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

**10.28.3.40 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

**10.28.3.41 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

**10.28.3.42 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

**10.28.3.43 int cSDHSerial::numaxis (void) throw (cSDHLibraryException\*)**

Return number of axis of SDH

**10.28.3.44 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.

**10.28.3.45 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.

**10.28.3.46 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

**10.28.3.47 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

**10.28.3.48 void SDH::cSDHSerial::Open (cSerialBase \* *\_com*) throw (cSDHLibraryException\*)**

Open the serial device and check connection to SDH by querying the SDH firmware version

**Parameters:**

*\_com* - 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.

#### **10.28.3.49 void SDH::cSDHSerial::Close () throw (cSDHLibraryException\*)**

Close connection to serial port.

#### **10.28.3.50 virtual bool SDH::cSDHSerial::IsOpen (void) [virtual]**

Return true if connection to SDH firmware/hardware is open

Implements [SDH::cSDHBase](#).

#### **10.28.3.51 void SDH::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.

#### **10.28.3.52 void SDH::cSDHSerial::ExtractFirmwareState () throw ( cSDHErrorCommunication\* )**

Try to extract the state of the SDH firmware from the last reply

#### **10.28.3.53 double SDH::cSDHSerial::GetDuration (char \* line) throw ( cSDHErrorCommunication\* )**

Return duration of the execution of a SDH command as reported by line

#### **10.28.3.54 double SDH::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

#### **10.28.3.55 void SDH::cSDHSerial::Sync () throw ( cSDHErrorCommunication\* )**

Read all pending lines from SDH to resync execution of PC and SDH.

**10.28.3.56 void SDH::cSDHSerial::SyncUnknown () throw ( cSDHErrorCommunication\* )**

Read an unknown number of lines from SDH to resync execution of PC and SDH.

**10.28.3.57 cSimpleVector SDH::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.

**10.28.3.58 cSimpleVector SDH::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](#)

**10.28.3.59 cSimpleVector SDH::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**

#### **10.28.3.60 cSimpleVector SDH::cSDHSerial::ilim (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.

#### **10.28.3.61 cSimpleVector SDH::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 single number or a boolean value then all axes power states are set to that value, the NUMBER\_OF\_AXES-list is returned.

#### **10.28.3.62 void SDH::cSDHSerial::demo (bool onoff)**

Enable/disable SCHUNK demo

#### **10.28.3.63 int SDH::cSDHSerial::property (char const \* propname, int value)**

Set named property

Valid propnames are:

- "user\_errors"
- "terminal"
- "debug"

**10.28.3.64 int SDH::cSDHSerial::user\_errors (int *value*)**

**10.28.3.65 int SDH::cSDHSerial::terminal (int *value*)**

**10.28.3.66 int SDH::cSDHSerial::debug (int *value*)**

**10.28.3.67 cSimpleVector SDH::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**

**10.28.3.68 cSimpleVector SDH::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.

**10.28.3.69 cSimpleVector SDH::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).

#### **10.28.3.70 cSimpleVector SDH::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.

#### **10.28.3.71 cSimpleVector SDH::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.

#### **10.28.3.72 double SDH::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

#### **10.28.3.73 void SDH::cSDHSerial::stop (void) throw (cSDHLibraryException\*)**

Stop sdh.

Will NOT interrupt a previous "selgrip" or "grip" command, only an "m" command!

**10.28.3.74 eVelocityProfile SDH::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.

**10.28.3.75 eControllerType SDH::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.

**10.28.3.76 cSimpleVector SDH::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.

**10.28.3.77 cSimpleVector SDH::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

---

**10.28.3.78 cSimpleVector SDH::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

**10.28.3.79 cSimpleVector SDH::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.

**10.28.3.80 cSimpleVector SDH::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.

**10.28.3.81 cSimpleVector SDH::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.

**10.28.3.82 cSimpleVector SDH::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.

**10.28.3.83 cSimpleVector SDH::cSDHSerial::temp\_electronics (void) throw (cSDHLibraryException\*)**

Get actual temperatures of the electronics, i.e. teh 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.

**10.28.3.84 char\* SDH::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

**10.28.3.85 char\* SDH::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

**10.28.3.86 char\* SDH::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

**10.28.3.87 char\* SDH::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

**10.28.3.88 char\* SDH::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

**10.28.3.89 char\* SDH::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

**10.28.3.90 int SDH::cSDHSerial::numaxis (void) throw (cSDHLibraryException\*)**

Return number of axis of SDH

**10.28.3.91 cSimpleVector SDH::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.

**10.28.3.92 cSimpleVector SDH::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.

**10.28.3.93 double SDH::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

**10.28.3.94 double SDH::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

## 10.28.4 Member Data Documentation

**10.28.4.1 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)

**10.28.4.2 char const\* SDH::cSDHSerial::EOL [protected]**

String to use as "End Of Line" marker when sending to SDH.

**10.28.4.3 cSerialBase\* SDH::cSDHSerial::com [protected]**

The communication object to the serial device (RS232 port or ESD CAN net).

**10.28.4.4 cSimpleStringList SDH::cSDHSerial::reply [protected]**

Space for the replies from the SDH.

**10.28.4.5 int SDH::cSDHSerial::nb\_lines\_to\_ignore [protected]**

number of remaining reply lines of a previous (non-sequential) command

**10.28.4.6 char const\* SDH::cSDHSerial::EOL [protected]**

String to use as "End Of Line" marker when sending to SDH.

**10.28.4.7 cSerialBase\* SDH::cSDHSerial::com [protected]**

The communication object to the serial device (RS232 port or ESD CAN net).

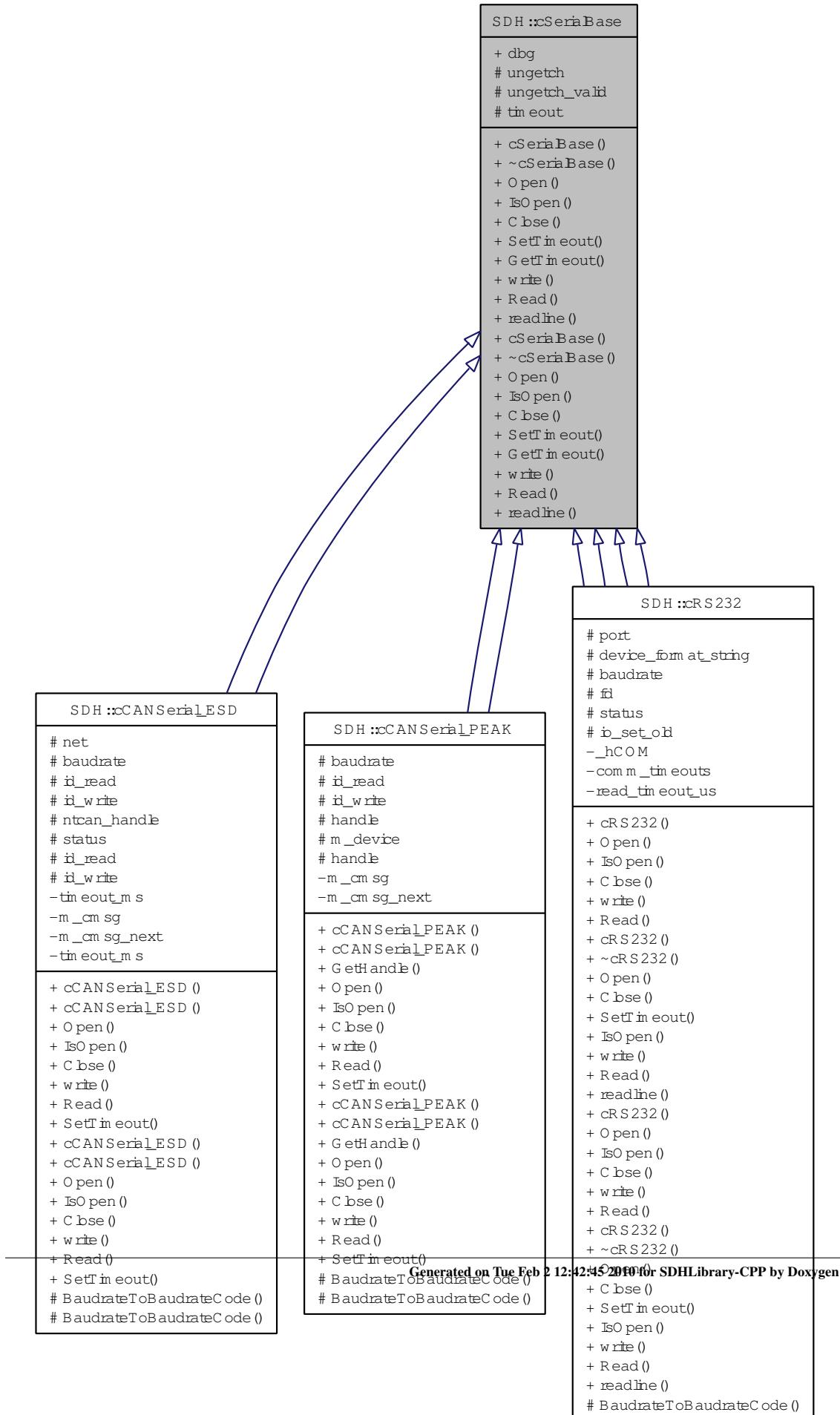
The documentation for this class was generated from the following files:

- cpp.desire.final/sdh/[sdhserial.h](#)
- sdh/[sdhserial.h](#)
- cpp.desire.final/sdh/[sdhserial.cpp](#)
- sdh/[sdhserial.cpp](#)

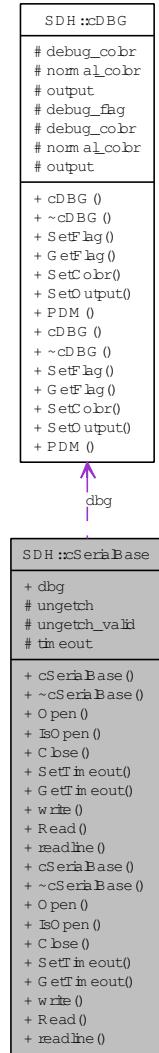
## **10.29 SDH::cSerialBase Class Reference**

```
#include <serialbase.h>
```

Inheritance diagram for SDH::cSerialBase:



Collaboration diagram for SDH::cSerialBase:



### 10.29.1 Detailed Description

Low-level communication class to access a serial port.

(This is an abstract base class with pure virtual functions)

#### 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** ()
 

*set 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.*
- **cSerialBase** ()
 

*ctor*
- **~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** ()
 

*set 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.*

## 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*

## 10.29.2 Constructor & Destructor Documentation

### 10.29.2.1 SDH::cSerialBase::cSerialBase () [inline]

ctor

### 10.29.2.2 virtual SDH::cSerialBase::~cSerialBase (void) [inline, virtual]

dtor

### 10.29.2.3 SDH::cSerialBase::cSerialBase () [inline]

ctor

### 10.29.2.4 virtual SDH::cSerialBase::~cSerialBase (void) [inline, virtual]

dtor

## 10.29.3 Member Function Documentation

### 10.29.3.1 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

#### **10.29.3.2 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

#### **10.29.3.3 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

#### **10.29.3.4 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), and [SDH::cRS232](#).

#### **10.29.3.5 virtual double SDH::cSerialBase::GetTimeout () [inline, virtual]**

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

#### **10.29.3.6 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:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

##### **Returns:**

the number of bytes actually written

Implemented in [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.7 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.8 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

**Parameters:**

*line* - ptr to where to store the read line

*size* - space available in line (bytes)

*eol* - a string containing all the chars that mark an end of line

*return\_on\_less\_data* - 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.

A pointer to the line read is returned.

**10.29.3.9 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.10 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.11 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

---

**10.29.3.12 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), and [SDH::cRS232](#).

**10.29.3.13 virtual double SDH::cSerialBase::GetTimeout () [inline, virtual]**

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

**10.29.3.14 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:**

*ptr* - pointer the byte array to send in memory

*len* - number of bytes to send

**Returns:**

the number of bytes actually written

Implemented in [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), [SDH::cRS232](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.15 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](#), [SDH::cCANSerial\\_ESD](#), [SDH::cCANSerial\\_PEAK](#), [SDH::cRS232](#), and [SDH::cRS232](#).

**10.29.3.16 virtual char\* SDH::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

**Parameters:**

*line* - ptr to where to store the read line

*size* - space available in line (bytes)

*eol* - a string containing all the chars that mark an end of line

*return\_on\_less\_data* - 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.

A pointer to the line read is returned.

## 10.29.4 Member Data Documentation

### 10.29.4.1 char SDH::cSerialBase::ungetch [protected]

an already read data byte of the next line

### 10.29.4.2 bool SDH::cSerialBase::ungetch\_valid [protected]

Flag, true if ungetch is valid.

### 10.29.4.3 double SDH::cSerialBase::timeout [protected]

timeout in seconds

### 10.29.4.4 cDBG SDH::cSerialBase::dbg

A stream object to print colored debug messages.

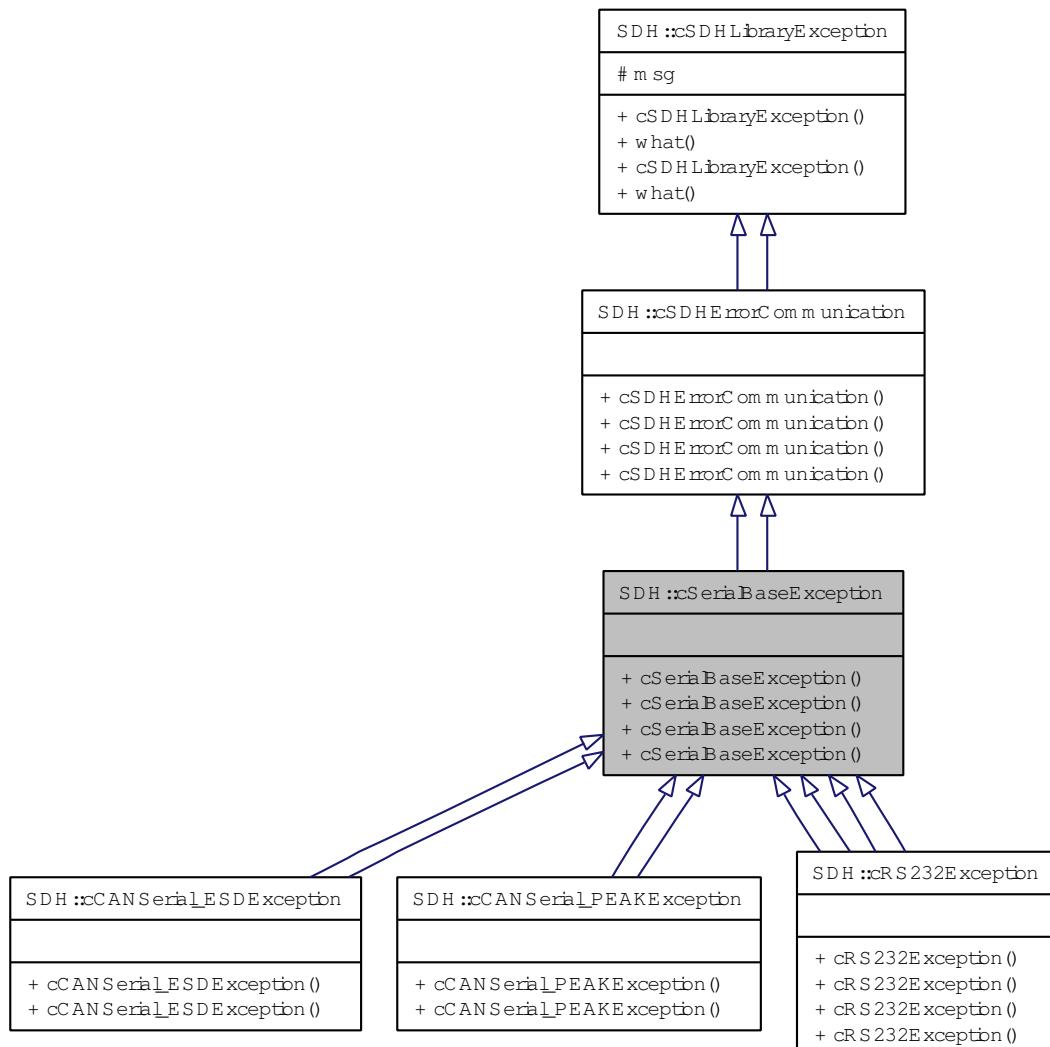
The documentation for this class was generated from the following files:

- [cpp.desire.final/sdh/serialbase.h](#)
- [sdh/serialbase.h](#)
- [cpp.desire.final/sdh/serialbase.cpp](#)
- [sdh/serialbase.cpp](#)

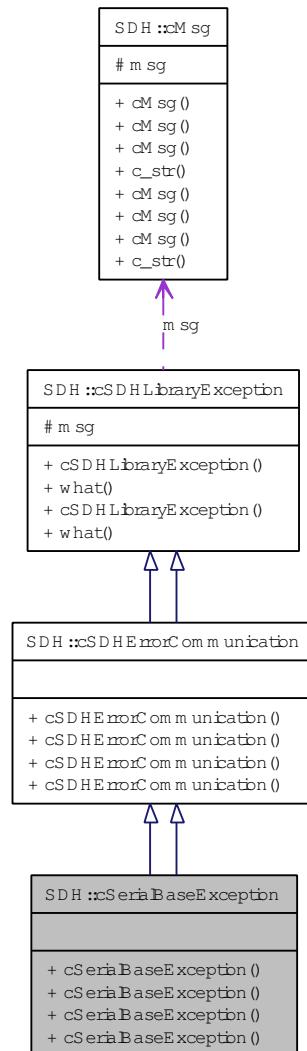
## 10.30 SDH::cSerialBaseException Class Reference

```
#include <serialbase.h>
```

Inheritance diagram for SDH::cSerialBaseException:



Collaboration diagram for SDH::cSerialBaseException:



### 10.30.1 Detailed Description

Derived exception class for low-level serial communication related exceptions.

#### Public Member Functions

- [cSerialBaseException \(cMsg const &\\_msg\)](#)
- [cSerialBaseException \(char const \\*\\_type, cMsg const &\\_msg\)](#)
- [cSerialBaseException \(cMsg const &\\_msg\)](#)
- [cSerialBaseException \(char const \\*\\_type, cMsg const &\\_msg\)](#)

## 10.30.2 Constructor & Destructor Documentation

**10.30.2.1 SDH::cSerialBaseException::cSerialBaseException (cMsg const & *\_msg*) [inline]**

**10.30.2.2 SDH::cSerialBaseException::cSerialBaseException (char const \* *\_type*, cMsg const & *\_msg*) [inline]**

**10.30.2.3 SDH::cSerialBaseException::cSerialBaseException (cMsg const & *\_msg*) [inline]**

**10.30.2.4 SDH::cSerialBaseException::cSerialBaseException (char const \* *\_type*, cMsg const & *\_msg*) [inline]**

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

- cpp.desire.final/sdh/[serialbase.h](#)
- sdh/[serialbase.h](#)

## 10.31 SDH::cSimpleStringList Class Reference

```
#include <simplestringlist.h>
```

### 10.31.1 Detailed Description

A simple string list. (Fixed maximum number of strings of fixed maximum length).

#### Public Types

- enum { **eMAX\_LINES** = 256, **eMAX\_CHARS** = 256 }  
*anonymous enum instead of define macros*
- 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*
- **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*

## 10.31.2 Member Enumeration Documentation

### 10.31.2.1 anonymous enum

anonymous enum instead of define macros

**Enumerator:**

**eMAX\_LINES**  
**eMAX\_CHARS**

### 10.31.2.2 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.

#### 10.31.3.2 SDH::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 char \* cSimpleStringList::NextLine ()

Return the next line, this increases current\_line.

#### 10.31.4.3 int cSimpleStringList::Length () const

Return number of lines stored.

#### 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

#### 10.31.4.7 char\* SDH::cSimpleStringList::CurrentLine ()

Return the current line.

#### 10.31.4.8 char\* SDH::cSimpleStringList::NextLine ()

Return the next line, this increases current\_line.

**10.31.4.9 int SDH::cSimpleStringList::Length () const**

Return number of lines stored.

**10.31.4.10 char\* SDH::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, ...

**10.31.4.11 char const\* SDH::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, ...

**10.31.4.12 void SDH::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 [protected]**

a fixed length array of lines with fixed length

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

- cpp.desire.final/sdh/[simplestringlist.h](#)
- sdh/[simplestringlist.h](#)
- cpp.desire.final/sdh/[simplestringlist.cpp](#)
- sdh/[simplestringlist.cpp](#)

## 10.32 SDH::cSimpleTime Class Reference

```
#include <simpletime.h>
```

### 10.32.1 Detailed Description

Very simple class to measure elapsed time.

#### 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.*
- **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`

### 10.32.2 Constructor & Destructor Documentation

#### 10.32.2.1 `SDH::cSimpleTime::cSimpleTime () [inline]`

Constructor: store current time ("now") internally.

#### 10.32.2.2 `SDH::cSimpleTime::cSimpleTime () [inline]`

Constructor: store current time ("now") internally.

### 10.32.3 Member Function Documentation

#### 10.32.3.1 `void SDH::cSimpleTime::StoreNow (void) [inline]`

Store current time internally.

#### 10.32.3.2 `double SDH::cSimpleTime::Elapsed (void) const [inline]`

Return time in seconds elapsed between the time stored in the object and now.

#### 10.32.3.3 `long SDH::cSimpleTime::Elapsed_us (void) const [inline]`

Return time in micro seconds elapsed between the time stored in the object and now.

#### 10.32.3.4 `double SDH::cSimpleTime::Elapsed (cSimpleTime const & other) const [inline]`

Return time in seconds elapsed between the time stored in the object and *other*.

#### 10.32.3.5 `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*.

#### 10.32.3.6 `timeval SDH::cSimpleTime::Timeval (void) [inline]`

Return the time stored as C struct timeval.

**10.32.3.7 void SDH::cSimpleTime::StoreNow (void) [inline]**

Store current time internally.

**10.32.3.8 double SDH::cSimpleTime::Elapsed (void) const [inline]**

Return time in seconds elapsed between the time stored in the object and now.

**10.32.3.9 long SDH::cSimpleTime::Elapsed\_us (void) const [inline]**

Return time in micro seconds elapsed between the time stored in the object and now.

**10.32.3.10 double SDH::cSimpleTime::Elapsed (cSimpleTime const & other) const [inline]**

Return time in seconds elapsed between the time stored in the object and *other*.

**10.32.3.11 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*.

**10.32.3.12 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 [read, protected]**

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

- cpp.desire.final/sdh/[simpletime.h](#)
- sdh/[simpletime.h](#)

## 10.33 SDH::cSimpleVector Class Reference

```
#include <simplevector.h>
```

### 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.

### Public Types

- enum { [eNUMBER\\_OF\\_ELEMENTS](#) = 7 }  
*anonymous enum (instead of define like macros)*
- 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.*
- 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 y = the second element, if that is valid.*
- double & [z](#) (void)  
*Interpret object as x/y/z vector: return z = the third 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).*

- `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.*
- `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*

## 10.33.2 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

### 10.33.2.2 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 SDH::cSimpleVector::cSimpleVector () throw (cSimpleVectorException\*)

Default constructor: init members to zero.

### 10.33.3.5 SDH::cSimpleVector::cSimpleVector (int *\_nb\_values*, char const \* *str*) throw (cSimpleVectorException\*)

Constructor: init members from *\_nb\_values* comma separated values in the give string *str*.

### 10.33.3.6 SDH::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*.

## 10.33.4 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 double & cSimpleVector::x (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.4 double & cSimpleVector::y (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.5 double & cSimpleVector::z (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.6 bool cSimpleVector::Valid (unsigned int index) const**

Return true if vector element *index* is valid (has been accessed at least once).

**10.33.4.7 void SDH::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*

**10.33.4.8 double& SDH::cSimpleVector::operator[ ] (unsigned int index)**

index operator, return a reference to the *index-th* element of this

**10.33.4.9 double& SDH::cSimpleVector::x (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.10 double& SDH::cSimpleVector::y (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.11 double& SDH::cSimpleVector::z (void)**

Interpret object as x/y/z vector: return x = the first element, if that is valid.

**10.33.4.12 bool SDH::cSimpleVector::Valid (unsigned int index) const**

Return true if vector element *index* is valid (has been accessed at least once).

### 10.33.5 Member Data Documentation

**10.33.5.1 double SDH::cSimpleVector::value [protected]**

**10.33.5.2 int SDH::cSimpleVector::valid [protected]**

bit mask which values in [value](#) are valid

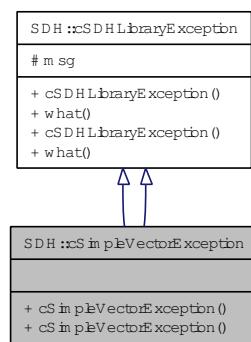
The documentation for this class was generated from the following files:

- [cpp.desire.final/sdh/simplevector.h](#)
- [sdh/simplevector.h](#)
- [cpp.desire.final/sdh/simplevector.cpp](#)
- [sdh/simplevector.cpp](#)

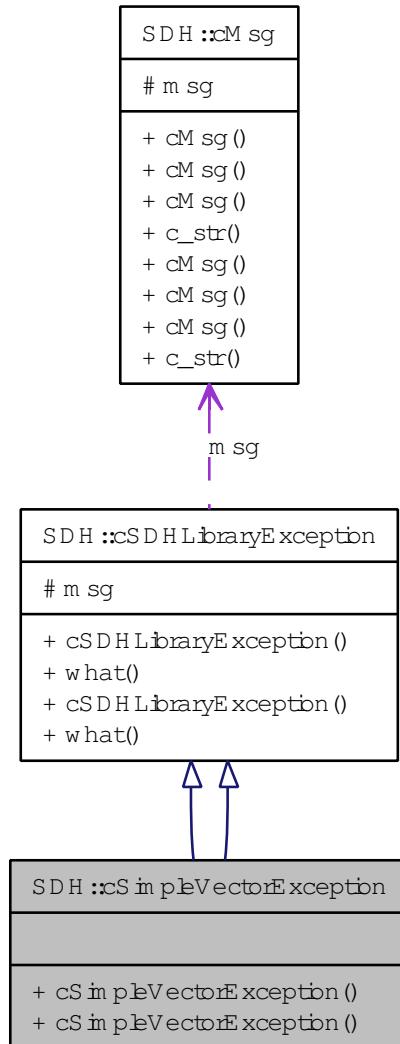
## 10.34 SDH::cSimpleVectorException Class Reference

```
#include <simplevector.h>
```

Inheritance diagram for SDH::cSimpleVectorException:



Collaboration diagram for SDH::cSimpleVectorException:



### 10.34.1 Detailed Description

Derived exception class for low-level simple vector related exceptions.

### Public Member Functions

- [cSimpleVectorException \(cMsg const &\\_msg\)](#)
- [cSimpleVectorException \(cMsg const &\\_msg\)](#)

## 10.34.2 Constructor & Destructor Documentation

**10.34.2.1 SDH::cSimpleVectorException::cSimpleVectorException (cMsg const & \_msg)  
[inline]**

**10.34.2.2 SDH::cSimpleVectorException::cSimpleVectorException (cMsg const & \_msg)  
[inline]**

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

- cpp.desire.final/sdh/[simplevector.h](#)
- sdh/[simplevector.h](#)

## 10.35 SDH::cUnitConverter Class Reference

```
#include <unit_converter.h>
```

### 10.35.1 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.

### 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.*

- `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.*
- 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")*

## 10.35.2 Constructor & Destructor Documentation

### 10.35.2.1 `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:

- `_kind` - a string describing the kind of unit to be converted (something like "angle" or "time")
- `_name` - the name of the external unit (something like "degrees" or "milliseconds")
- `_symbol` - the symbol of the external unit (something like "deg" or "ms")
- `_factor` - the conversion factor from internal to external units
- `_offset` - the conversion offset from internal to external units
- `_decimal_places` - 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

### 10.35.2.2 `SDH::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:

- `_kind` - a string describing the kind of unit to be converted (something like "angle" or "time")
- `_name` - the name of the external unit (something like "degrees" or "milliseconds")
- `_symbol` - the symbol of the external unit (something like "deg" or "ms")
- `_factor` - the conversion factor from internal to external units
- `_offset` - the conversion offset from internal to external units
- `_decimal_places` - 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

## 10.35.3 Member Function Documentation

### 10.35.3.1 `double cUnitConverter::ToExternal (double internal) const`

Convert single value *internal* given in internal units into external units. Returns *internal* \* `factor` + `offset`

**10.35.3.2 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.

**10.35.3.3 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**

**10.35.3.4 double cUnitConverter::ToInternal (double *external*) const**

Convert value *external* given in external units into internal units. Returns (*external* - **offset**) / **factor**

**10.35.3.5 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.

**10.35.3.6 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**

**10.35.3.7 char const \* cUnitConverter::GetKind (void) const**

Return the kind of unit converted (something like "angle" or "time").

**10.35.3.8 char const \* cUnitConverter::GetName (void) const**

Return the name of the external unit (something like "degrees" or "milliseconds").

**10.35.3.9 char const \* cUnitConverter::GetSymbol (void) const**

Return the symbol of the external unit (something like "deg" or "ms").

**10.35.3.10 double cUnitConverter::GetFactor (void) const**

Return the conversion factor from internal to external units.

**10.35.3.11 double cUnitConverter::GetOffset (void) const**

Return the conversion offset from internal to external units.

**10.35.3.12 int cUnitConverter::GetDecimalPlaces (void) const**

Return the number of decimal places for printing values in the external unit system.

**10.35.3.13 double SDH::cUnitConverter::ToExternal (double *internal*) const**

Convert single value *internal* given in internal units into external units. Returns *internal* \* **factor** + **offset**

**10.35.3.14 cSimpleVector SDH::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.

**10.35.3.15 std::vector<double> SDH::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**

**10.35.3.16 double SDH::cUnitConverter::ToInternal (double *external*) const**

Convert value *external* given in external units into internal units. Returns (*external* - **offset**) / **factor**

**10.35.3.17 cSimpleVector SDH::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.

**10.35.3.18 std::vector<double> SDH::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**

**10.35.3.19 char const\* SDH::cUnitConverter::GetKind (void) const**

Return the kind of unit converted (something like "angle" or "time").

**10.35.3.20 char const\* SDH::cUnitConverter::GetName (void) const**

Return the name of the external unit (something like "degrees" or "milliseconds").

**10.35.3.21 char const\* SDH::cUnitConverter::GetSymbol (void) const**

Return the symbol of the external unit (something like "deg" or "ms").

**10.35.3.22 double SDH::cUnitConverter::GetFactor (void) const**

Return the conversion factor from internal to external units.

**10.35.3.23 double SDH::cUnitConverter::GetOffset (void) const**

Return the conversion offset from internal to external units.

**10.35.3.24 int SDH::cUnitConverter::GetDecimalPlaces (void) const**

Return the number of decimal places for printing values in the external unit system.

## 10.35.4 Member Data Documentation

**10.35.4.1 char const\* SDH::cUnitConverter::kind [protected]**

the kind of unit to be converted (something like "angle" or "time")

**10.35.4.2 char const\* SDH::cUnitConverter::name [protected]**

the name of the external unit (something like "degrees" or "milliseconds")

**10.35.4.3 char const\* SDH::cUnitConverter::symbol [protected]**

the symbol of the external unit (something like "deg" or "ms")

**10.35.4.4 double SDH::cUnitConverter::factor [protected]**

the conversion factor from internal to external units

**10.35.4.5 double SDH::cUnitConverter::offset [protected]**

the conversion offset from internal to external units

**10.35.4.6 int SDH::cUnitConverter::decimal\_places [protected]**

A usefull number of decimal places for printing values in the external unit system.

**10.35.4.7 char const\* SDH::cUnitConverter::kind [protected]**

the kind of unit to be converted (something like "angle" or "time")

**10.35.4.8 char const\* SDH::cUnitConverter::name [protected]**

the name of the external unit (something like "degrees" or "milliseconds")

**10.35.4.9 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:

- [cpp.desire.final/sdh/unit\\_converter.h](#)
- [sdh/unit\\_converter.h](#)
- [cpp.desire.final/sdh/unit\\_converter.cpp](#)
- [sdh/unit\\_converter.cpp](#)

## 10.36 option Struct Reference

```
#include <getopt.h>
```

### Public Attributes

- `char * name`
- `int has_arg`
- `int * flag`
- `int val`
- `char * name`
- `int * flag`

### 10.36.1 Member Data Documentation

#### 10.36.1.1 `char* option::name`

#### 10.36.1.2 `int option::has_arg`

#### 10.36.1.3 `int* option::flag`

#### 10.36.1.4 `int option::val`

#### 10.36.1.5 `char* option::name`

#### 10.36.1.6 `int* option::flag`

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

- `cpp.desire.final/vcc/getopt.h`
- `vcc/getopt.h`



# Chapter 11

## File Documentation

### 11.1 architecture.dox File Reference

#### 11.1.1 Detailed Description

Short overview of the SDHLibrary-CPP and [SDH](#) architecture.

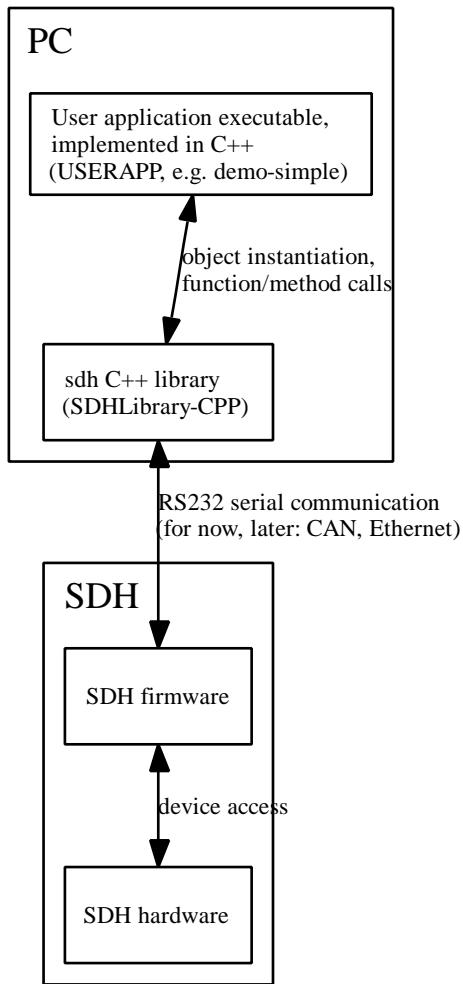
#### 11.1.2 Overview

##### 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 == "COM1"
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 cpp.desire.final/architecture.dox File Reference

### 11.2.1 Detailed Description

Short overview of the SDHLibrary-CPP and [SDH](#) architecture.

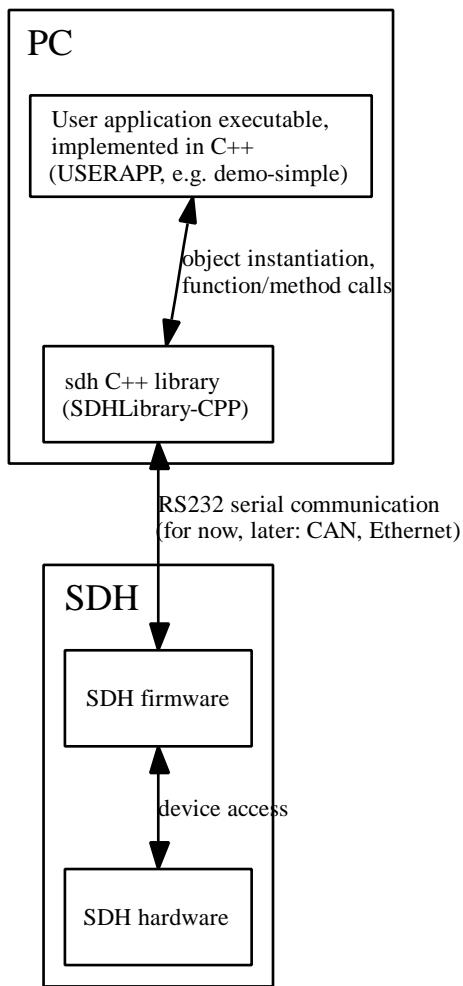
### 11.2.2 Overview

#### 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 == "COM1"
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

## 11.3 connectors.dox File Reference

### 11.3.1 Detailed Description

### 11.3.2 General project information

**Author:**

Dirk Osswald

**Date:**

2007-02-19

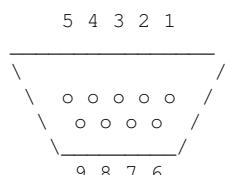
### 11.3.3 Purpose

This page describes the connectors of the [SDH](#) / LWA

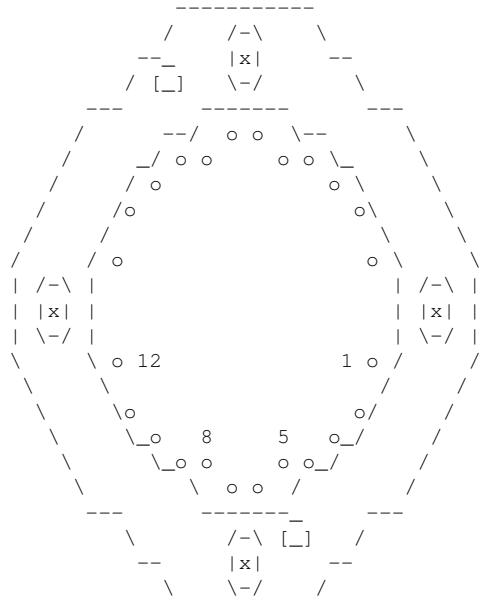
**SDH**

RS232 Communication connector cable of the [SDH](#) at the base of the LWA:

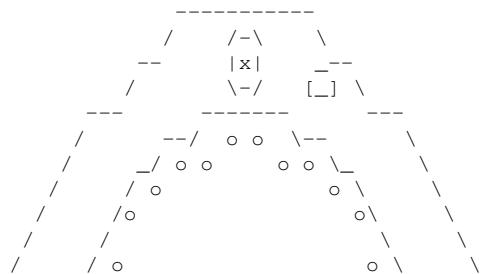
- 9pin Sub-D female connector (View from connecting side):

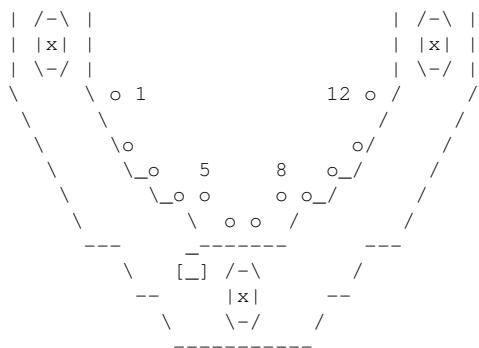


- 1 : here: nc, (normally: Data Carrier Detect)
- 2 : RxD of PC = TxD of [SDH](#)
- 3 : TxD 0f PC = RxD of [SDH](#)
- 4 : here: nc, (normally: Data Terminal Ready)
- 5 : GND
- 6 : here: nc, (normally: Data Set Ready)
- 7 : here: nc, (normally: Request To Send)
- 8 : here: nc, (normally: Clear To Send)
- 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](#)



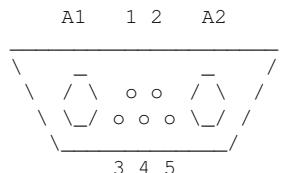


- 1 : Pwr (RD) 24V supply for power and logic of SDH2
  - 2 : TxD (BN) RS232 TxD of PC = RxD of SDH2
  - 3 : RxD (WH) RS232 RxD of PC = TxD of SDH2
  - 4 : TxD2 (RD) RS232 TxD of PC = RxD of SDH2, second RS232 channel of SDH2
  - 5 : RxD2 (PK) RS232 RxD of PC = TxD of SDH2, second RS232 channel of SDH2
  - 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

LWA

Combined power / CAN connector of the LWA:

- Male connector (View from connecting side):



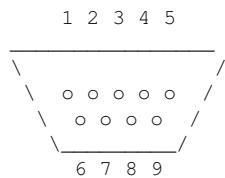
- A1 : +24V
  - A2 : 0V (blue)
  - 1 : CAN-High (Profibus-A1) (green)
  - 2 : RxD

- 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

#### 11.3.4 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

## 11.4 cpp.desire.final/connectors.dox File Reference

### 11.4.1 Detailed Description

### 11.4.2 General project information

**Author:**

Dirk Osswald

**Date:**

2007-02-19

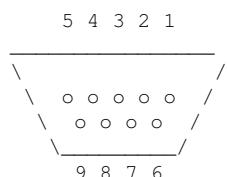
### 11.4.3 Purpose

This page describes the connectors of the [SDH](#) / LWA

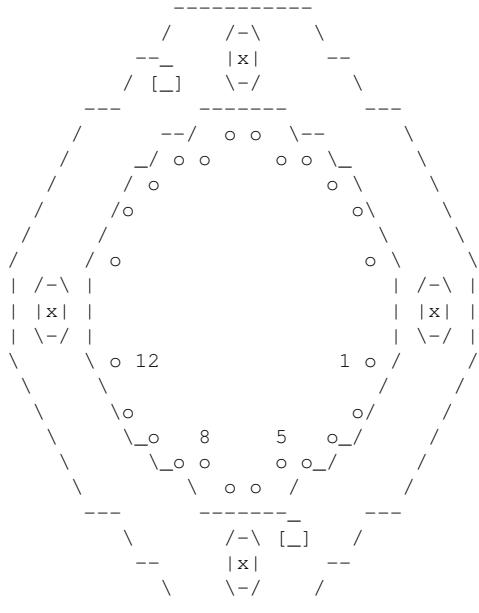
#### SDH

RS232 Communication connector cable of the [SDH](#) at the base of the LWA:

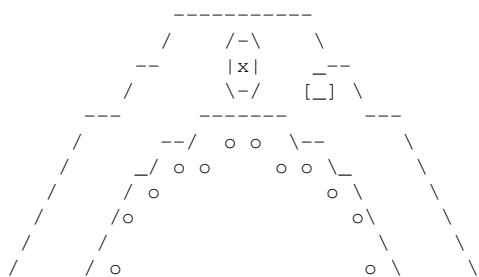
- 9pin Sub-D female connector (View from connecting side):

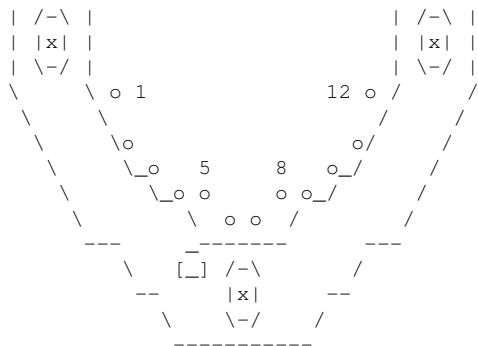


- 1 : here: nc, (normally: Data Carrier Detect)
- 2 : RxD of PC = TxD of [SDH](#)
- 3 : TxD 0f PC = RxD of [SDH](#)
- 4 : here: nc, (normally: Data Terminal Ready)
- 5 : GND
- 6 : here: nc, (normally: Data Set Ready)
- 7 : here: nc, (normally: Request To Send)
- 8 : here: nc, (normally: Clear To Send)
- 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 : TxDS2 (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](#)



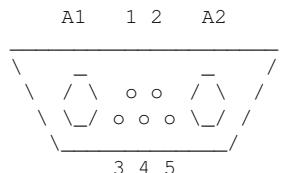


- 1 : Pwr (RD) 24V supply for power and logic of SDH2
  - 2 : TxD (BN) RS232 TxD of PC = RxD of SDH2
  - 3 : RxD (WH) RS232 RxD of PC = TxD of SDH2
  - 4 : Tx D2 (RD) RS232 TxD of PC = RxD of SDH2, second RS232 channel of SDH2
  - 5 : Rx D2 (PK) RS232 RxD of PC = TxD of SDH2, second RS232 channel of 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

LWA

Combined power / CAN connector of the LWA:

- Male connector (View from connecting side):



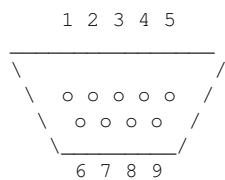
- A1 : +24V
  - A2 : 0V (blue)
  - 1 : CAN-High (Profibus-A1) (green)
  - 2 : RxD

- 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

#### 11.4.4 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

## 11.5 `cpp.desire.final/demo/cancat.cpp` File Reference

### **11.5.1 Detailed Description**

Yet incomplete tool to send and receive data via CAN. See [help](#) and online help ("`-h`" or "`--help`") for available options.

### **11.5.2 General file information**

**Author:**

Dirk Osswald

Date:

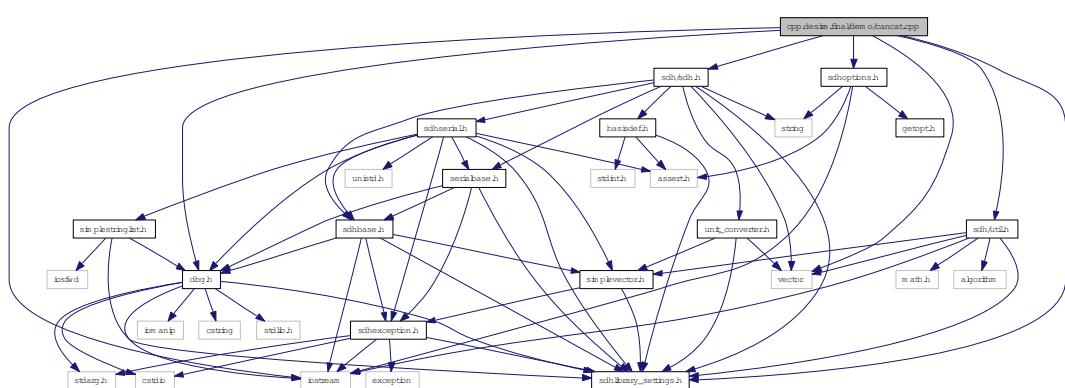
2007-01-18

### 11.5.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dbg.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "adoptions.h"
```

Include dependency graph for cancat.cpp:



## Functions

- int main (int argc, char \*\*argv)

## Variables

### Some informative variables

- char const \* **help** = "Send data from command line via ESD or PEAK CAN and display replies until CTRL-C is pressed."
- char const \* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **url** = "http://www.schunk.com"
- char const \* **version** = "\$Id: cancat.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **copyright** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.5.4 Function Documentation

#### 11.5.4.1 int main (int *argc*, char \*\* *argv*)

### 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** = "Send data from command line via ESD or PEAK CAN and display replies until CTRL-C is pressed."

11.5.5.4 char const\* **url** = "http://www.schunk.com"

11.5.5.5 char const\* **version** = "\$Id: cancat.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.6 demo/cancat.cpp File Reference

### 11.6.1 Detailed Description

Yet incomplete tool to send and receive data via CAN. See [\\_help\\_](#) and online help (" -h " or " -help ") for available options.

### 11.6.2 General file information

#### Author:

Dirk Osswald

#### Date:

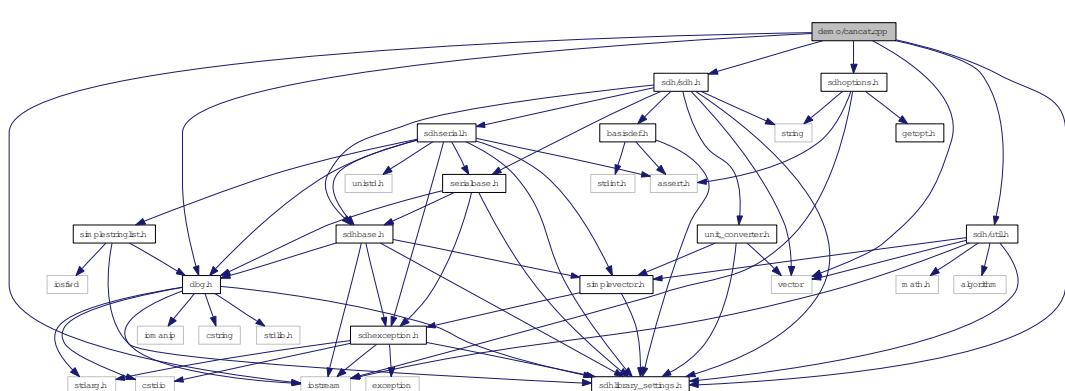
2007-01-18

### 11.6.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dbg.h"
#include "sdh/util.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdhoptions.h"
```

Include dependency graph for cancat.cpp:



## Functions

- int `main` (int argc, char \*\*argv)

## Variables

### Some informative variables

- char const \* **help** = "Send data from command line via ESD or PEAK CAN and display replies until CTRL-C is pressed."
- char const \* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **url** = "http://www.schunk.com"
- char const \* **version** = "\$Id: cancat.cpp 5000 2009-12-01 10:41:18Z Osswald2 \$"
- char const \* **copyright** = "Copyright (c) 2007 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) 2007 SCHUNK GmbH & Co. KG"

11.6.5.3 char const\* **help** = "Send data from command line via ESD or PEAK CAN and display replies until CTRL-C is pressed."

11.6.5.4 char const\* **url** = "http://www.schunk.com"

11.6.5.5 char const\* **version** = "\$Id: cancat.cpp 5000 2009-12-01 10:41:18Z Osswald2 \$"

## 11.7 cpp.desire.final/demo/demo-contact-grasping.cpp File Reference

### 11.7.1 Detailed Description

Simple script to do grasping using tactile sensor info feedback. See [\\_help\\_](#) and online help ("`-h`" or "`--help`") for available options.

### 11.7.2 General file information

#### Author:

Dirk Osswald

#### Date:

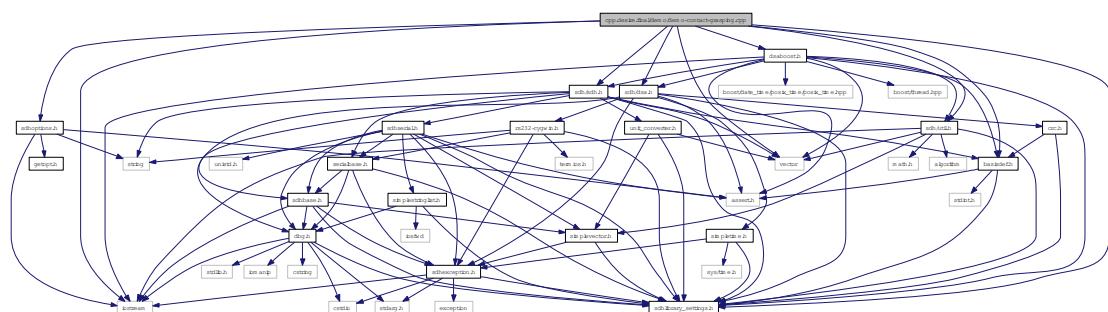
2009-08-02

### 11.7.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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

- 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`
- `cDBG cdbg` (false,"red")

### Some informative variables

- char const \* `_help_`
- 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 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.7.4 Function Documentation

**11.7.4.1 void AxisAnglesToFingerAngles (std::vector< double > aa, std::vector< double > &fa0, std::vector< double > &fa1, std::vector< double > &fa2)**

**11.7.4.2 void GotoStartPose (cSDH & hand, char const \* msg)**

**11.7.4.3 int main (int argc, char \*\* argv)**

finished:

## 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) 2007 SCHUNK GmbH & Co. KG"**

**11.7.5.3 char const\* `_help_`**

**Initial value:**

```
"Simple script to do grasping with tactile sensor info feedback:\n"
"The hand will move to a pregrasp pose (open hand). You can then\n"
"reach an object to grasp into the hand. The actual grasping\n"
"is started as soon as a contact is detected. The finger\n"
"joints then try to move inwards until a certain\n"
"force is reached on the corresponding tactile sensors.\n"
"\n"
"- Example usage:\n"
"- Make SDH connected to port 2 = COM3 with tactile sensors:\n"
"  connected to port 3 = COM4 grasp:\n"
"  > demo-contact-grasping -p 2 --dsaport=3\n"
"  \n"
"- Make SDH connected to USB to RS232 converter 0 and with tactile sensors:\n"
"  connected to USB to RS232 converter 1 grasp:\n"
"  > demo-contact-grasping --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/ttyUSB0\n"
```

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

**11.7.5.4 char const\* \_\_url\_\_ = "http://www.schunk.com"**

**11.7.5.5 char const\* \_\_version\_\_ = "\$Id: demo-contact-grasping.cpp 4932 2009-11-02 08:20:24Z  
Osswald2 \$"**

**11.7.5.6 cDBG cdbg(false,"red")**

**11.7.5.7 char const\* usage**

**Initial value:**

```
"usage: demo-contact-grasping [options]\n"
```

## 11.8 demo/demo-contact-grasping.cpp File Reference

### 11.8.1 Detailed Description

Simple script to do grasping using tactile sensor info feedback. See [\\_help\\_](#) and online help (" -h" or " -help") for available options.

### 11.8.2 General file information

#### Author:

Dirk Osswald

#### Date:

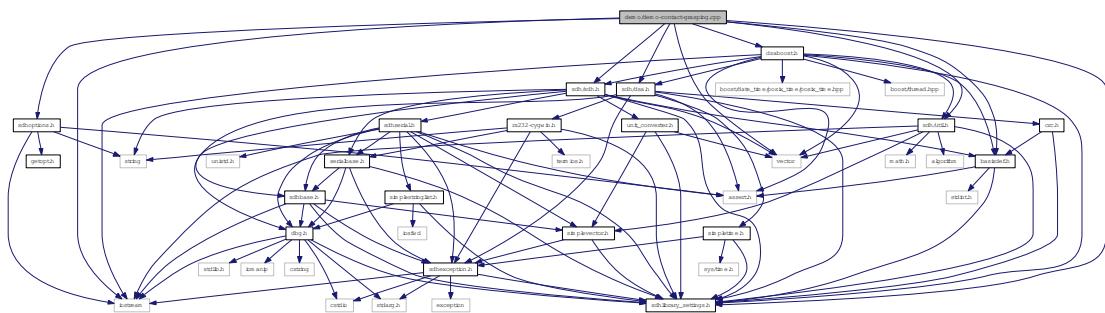
2009-08-02

### 11.8.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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 "dsaboostr.h"
```

Include dependency graph for demo-contact-grasping.cpp:



## Functions

- 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**
- **cDBG cdbg** (false, "red")

### Some informative variables

- char const \* **\_help\_**
- 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 5022 2009-12-04 16:05:53Z Osswald2 \$"
- char const \* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.8.4 Function Documentation

**11.8.4.1 void AxisAnglesToFingerAngles (std::vector< double > aa, std::vector< double > & fa0, std::vector< double > & fa1, std::vector< double > & fa2)**

**11.8.4.2 void GotoStartPose (cSDH & hand, char const \* msg)**

**11.8.4.3 int main (int argc, char \*\* argv)**

finished:

## 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\_\_**

**Initial value:**

```
"Simple script to do grasping with tactile sensor info feedback:\n"
"The hand will move to a pregrasp pose (open hand). You can then\n"
"reach an object to grasp into the hand. The actual grasping\n"
"is started as soon as a contact is detected. The finger\n"
"joints then try to move inwards until a certain\n"
"force is reached on the corresponding tactile sensors.\n"
"\n"
"- Example usage:\n"
"- Make SDH connected to port 2 = COM3 with tactile sensors:\n"
"- connected to port 3 = COM4 grasp:\n"
"> demo-contact-grasping -p 2 --dsaport=3\n"
"\n"
"- Make SDH connected to USB to RS232 converter 0 and with tactile sensors:\n"
"- connected to USB to RS232 converter 1 grasp:\n"
"> demo-contact-grasping --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/ttyUSB0\n"
"\n"
- Get the version info of both the joint controllers and the tactile\n"
"sensor firmware from an SDH connected to:\n"
```

```
"      - port 2 = COM3 (joint controllers) and \n"
"      - port 3 = COM4 (tactile sensor controller) \n"
"      > demo-contact-grasping --port=2 --dsaport=3 -v\n"
```

**11.8.5.4 char const\* \_\_url\_\_ = "http://www.schunk.com"**

**11.8.5.5 char const\* \_\_version\_\_ = "\$Id: demo-contact-grasping.cpp 5022 2009-12-04 16:05:53Z  
Osswald2 \$"**

**11.8.5.6 cDBG cdbg(false,"red")**

**11.8.5.7 char const\* usage**

**Initial value:**

```
"usage: demo-contact-grasping [options]\n"
```

## 11.9 `cpp.desire.final/demo/demo-dsa.cpp` File Reference

### **11.9.1 Detailed Description**

Simple program to test class cDSA. See [help](#) and online help (" -h " or " -help ") for available options.

### **11.9.2 General file information**

**Author:**

Dirk Osswald

Date:

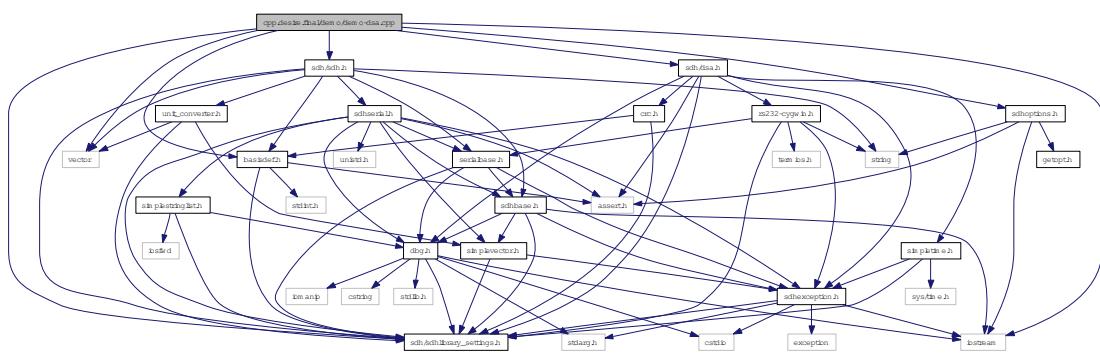
2008-06-12

### 11.9.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdh/sdhlibrary_settings.h"
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dsa.h"
#include "sdh/basisdef.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\_\_**
- char const \* **\_\_author\_\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **\_\_url\_\_** = "http://www.schunk.com"
- char const \* **\_\_version\_\_** = "\$Id: demo-dsa.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **\_\_copyright\_\_** = "Copyright (c) 2008 SCHUNK GmbH & Co. KG"
- char const \* **usage**

## 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) 2008 SCHUNK GmbH & Co. KG"

### 11.9.5.3 char const\* **\_\_help\_\_**

**Initial value:**

```
"Simple demo to test cDSA class of SDHLibrary-cpp.\n"
"\n"
"Remarks:\n"
"-- You must specify at least one of these options to see some output:\n"
"  -f | --fullframe  \n"
"  -r | --resulting\n"
"  -c | --controllerinfo \n"
"  -s | --sensorinfoinfo\n"
"  -m | --matrixinfo=N\n"
"\n"
"-- Example usage:\n"
"  - Read a single full frame from tactile sensors connected to port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -f\n"
"    \n"
"  - Read full frames continuously from tactile sensors connected to\n"
"    port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -f -r 1\n"
"    \n"
"  - Read a single full frame from tactile sensors connected to USB\n"
"    to RS232 converter 0:\n"
"    > demo-dsa --dsa_rs_device=/dev/ttyUSB0 -f \n"
"    \n"
"  - Read the sensor, controller, matrix 0 infos \n"
"    from tactile sensors connected to port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -s -c -m 0 \n"
"    \n"
"  - Get the version info of both the joint controllers and the tactile \n"
"    sensor firmware from an SDH connected to \n"
"    - port 2 = COM3 (joint controllers) and \n"
"    - port 3 = COM4 (tactile sensor controller) \n"
"    > demo-dsa -p 2 --dsaport=3 -v\n"
"\n"
"-- Known bugs:"
"  - see the bug description for \"cDSAEexception: Checksum Error on Windows\n"
```

```
"      console\" in the Related Pages->Bug List section of the doxygen\n"
"      documentation\n"
```

#### 11.9.5.4 **char const\* \_\_url\_\_ = "http://www.schunk.com"**

#### 11.9.5.5 **char const\* \_\_version\_\_ = "\$Id: demo-dsa.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"**

#### 11.9.5.6 **char const\* usage**

##### **Initial value:**

```
"usage: demo-dsa [options]\n"
```

## 11.10 demo/demo-dsa.cpp File Reference

### **11.10.1 Detailed Description**

Simple program to test class cDSA. See [help](#) and online help ("h" or "--help") for available options.

### **11.10.2 General file information**

**Author:**

Dirk Osswald

Date:

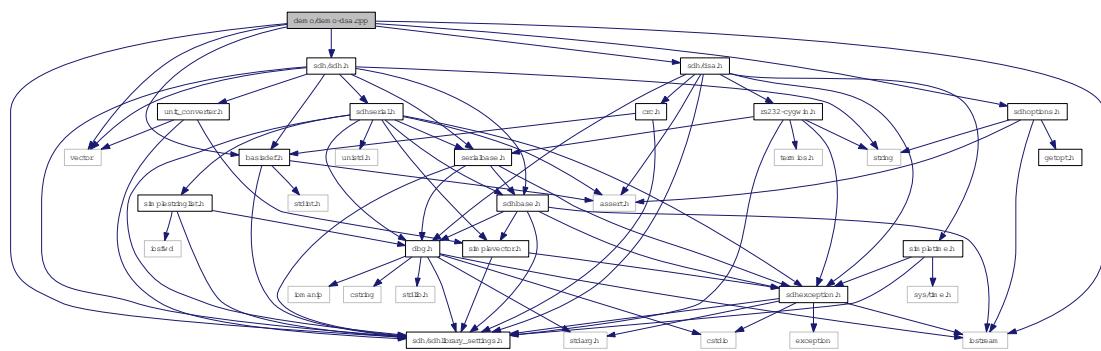
2008-06-12

### 11.10.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdh/sdhlibrary_settings.h"
#include <iostream>
#include <vector>
#include "sdh/sdh.h"
#include "sdh/dsa.h"
#include "sdh/basisdef.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](#)
- char const \* [author](#) = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* [url](#) = "http://www.schunk.com"
- char const \* [version](#) = "\$Id: demo-dsa.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* [copyright](#) = "Copyright (c) 2008 SCHUNK GmbH & Co. KG"
- char const \* [usage](#)

## 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) 2008 SCHUNK GmbH & Co. KG"

### 11.10.5.3 char const\* [help](#)

**Initial value:**

```
"Simple demo to test cDSA class of SDHLibrary-cpp.\n"
"\n"
"Remarks:\n"
"-- You must specify at least one of these options to see some output:\n"
"  -f | --fullframe  \n"
"  -r | --resulting\n"
"  -c | --controllerinfo \n"
"  -s | --sensorinfoinfo\n"
"  -m | --matrixinfo=N\n"
"\n"
"-- Example usage:\n"
"  - Read a single full frame from tactile sensors connected to port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -f\n"
"    \n"
"  - Read full frames continuously from tactile sensors connected to\n"
"    port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -f -r 1\n"
"    \n"
"  - Read a single full frame from tactile sensors connected to USB\n"
"    to RS232 converter 0:\n"
"    > demo-dsa --dsa_rs_device=/dev/ttyUSB0 -f \n"
"    \n"
"  - Read the sensor, controller, matrix 0 infos \n"
"    from tactile sensors connected to port 3 = COM4:\n"
"    > demo-dsa --dsaport=3 -s -c -m 0 \n"
"    \n"
"  - Get the version info of both the joint controllers and the tactile \n"
"    sensor firmware from an SDH connected to \n"
"    - port 2 = COM3 (joint controllers) and \n"
"    - port 3 = COM4 (tactile sensor controller) \n"
"    > demo-dsa -p 2 --dsaport=3 -v\n"
"\n"
"-- Known bugs:"
"  - see the bug description for \"cDSAEexception: Checksum Error on Windows\n"
```

```
"      console\" in the Related Pages->Bug List section of the doxygen\n"
"      documentation\n"
```

#### 11.10.5.4 **char const\* \_\_url\_\_ = "http://www.schunk.com"**

#### 11.10.5.5 **char const\* \_\_version\_\_ = "\$Id: demo-dsa.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"**

#### 11.10.5.6 **char const\* usage**

##### **Initial value:**

```
"usage: demo-dsa [options]\n"
```

## 11.11 [cpp.desire.final/demo/demo-GetAxisActualAngle.cpp](#) File Reference

### **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:

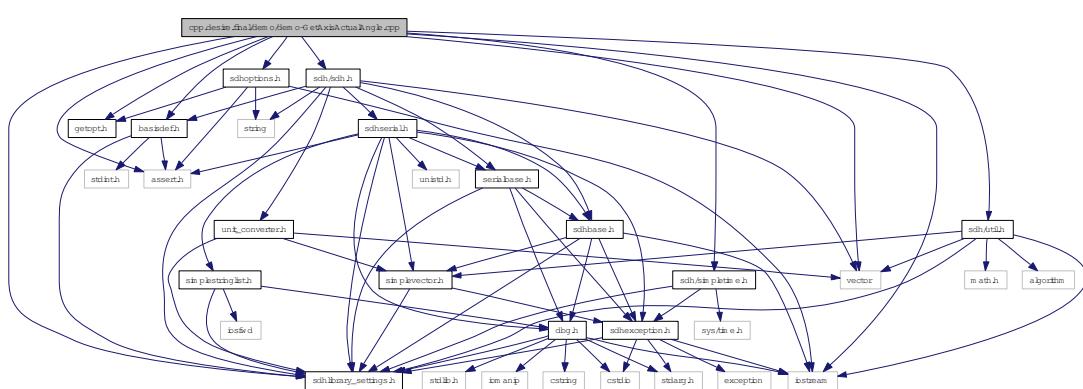
2007-03-07

### 11.11.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### Some informative variables

- char const \* **\_help\_**
- char const \* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **\_url\_** = "http://www.schunk.com"
- char const \* **\_version\_** = "\$Id: demo-GetAxisActualAngle.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.11.4 Function Documentation

### 11.11.4.1 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\_**

#### Initial value:

```
"Print measured actual axis angles of SDH.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
" - Print actual angles of an SDH connected to port 2 = COM3 once:\n"
"   > demo-GetAxisActualAngle -p 2\n"
"   \n"
" - Print actual angles of an SDH connected to port 2 = COM3 every 500ms:\n"
"   > demo-GetAxisActualAngle -p 2 -t 0.5\n"
"   \n"
" - Print actual angles of an SDH connected to USB to RS232 converter 0 once:\n"
"   > demo-GetAxisActualAngle --sdh_rs_device=/dev/ttyUSB0 \n"
"   \n"
" - Get the version info of an SDH connected to port 2 = COM3 \n"
"   > demo-GetAxisActualAngle --port=2 -v\n"
```

### 11.11.5.4 char const\* **\_url\_** = "http://www.schunk.com"

### 11.11.5.5 char const\* **\_version\_** = "\$Id: demo-GetAxisActualAngle.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.11.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-GetAxisActualAngle [options] \n"
```

## 11.12 demo/demo-GetAxisActualAngle.cpp File Reference

### **11.12.1 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.

### **11.12.2 General file information**

**Author:**

Dirk Osswald

Date:

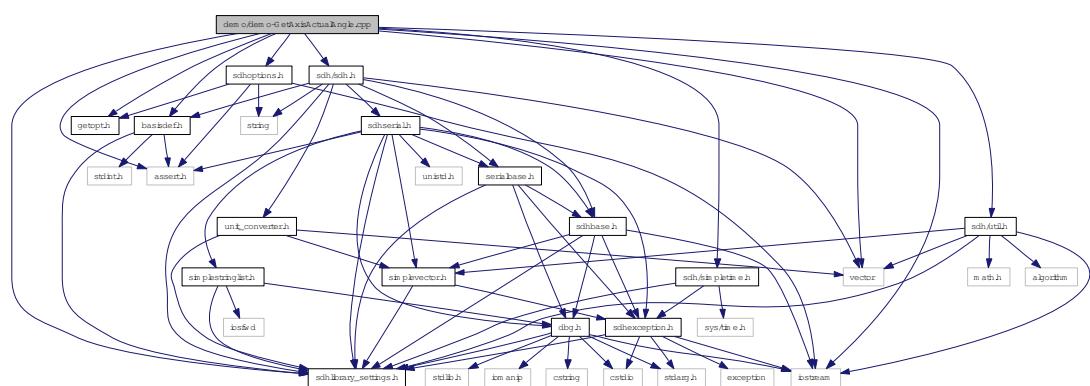
2007-03-07

### 11.12.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### Some informative variables

- char const \* **\_help\_**
- char const \* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **\_url\_** = "http://www.schunk.com"
- char const \* **\_version\_** = "\$Id: demo-GetAxisActualAngle.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **\_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\_**

#### Initial value:

```
"Print measured actual axis angles of SDH.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
" - Print actual angles of an SDH connected to port 2 = COM3 once:\n"
"   > demo-GetAxisActualAngle -p 2\n"
"   \n"
" - Print actual angles of an SDH connected to port 2 = COM3 every 500ms:\n"
"   > demo-GetAxisActualAngle -p 2 -t 0.5\n"
"   \n"
" - Print actual angles of an SDH connected to USB to RS232 converter 0 once:\n"
"   > demo-GetAxisActualAngle --sdh_rs_device=/dev/ttyUSB0 \n"
"   \n"
" - Get the version info of an SDH connected to port 2 = COM3 \n"
"   > demo-GetAxisActualAngle --port=2 -v\n"
```

### 11.12.5.4 char const\* **\_url\_** = "http://www.schunk.com"

### 11.12.5.5 char const\* **\_version\_** = "\$Id: demo-GetAxisActualAngle.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.12.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-GetAxisActualAngle [options] \n"
```

## 11.13 cpp.desire.final/demo/demo-GetFingerXYZ.cpp File Reference

### 11.13.1 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.

### 11.13.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-03-07

#### Bug

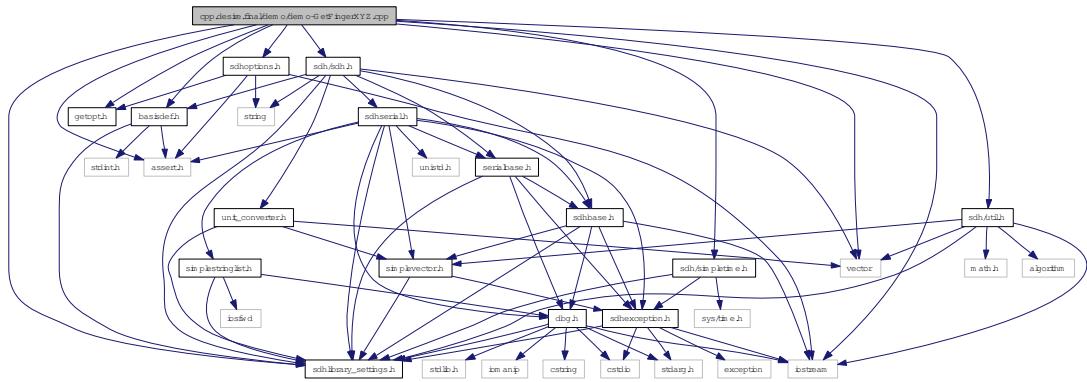
When compiled with MS Visual Studio then using the "-R" command line parameter will make the demonstration program demo-GetFingerXYZ abort.

### 11.13.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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

## Some informative variables

- char const \* help
  - char const \* author = "Dirk Osswald: dirk.osswald@de.schunk.com"
  - char const \* url = "http://www.schunk.com"
  - char const \* version = "\$Id: demo-GetFingerXYZ.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
  - char const \* 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\_\_**

## Initial value:

```
"Print measured XYZ position of fingertips of SDH.\n"
"C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"For every finger the actual axis angles and the finger tip coordinates\n"
"are printed.\n"
```

```
"\n"
"- Example usage:\n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to port 2 = COM3 once:\n"
"-       > demo-GetFingerXYZ -p 2\n"
"-     \n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to port 2 = COM3 every 500ms:\n"
"-       > demo-GetFingerXYZ -p 2 -t 0.5\n"
"-     \n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to USB to RS232 converter 0 once:\n"
"-       > demo-GetFingerXYZ --sdh_rs_device=/dev/ttyUSB0 \n"
"-     \n"
"-   - Get the version info of an SDH connected to port 2 = COM3 \n"
"-       > demo-GetFingerXYZ --port=2 -v\n"
"- Known bugs:\n"
"-   - Command line parameter \"-R\" does not work when compiled \n"
"-     with MS Visual Studio\n"
```

#### 11.13.5.4 char const\* \_\_url\_\_ = "http://www.schunk.com"

#### 11.13.5.5 char const\* \_\_version\_\_ = "\$Id: demo-GetFingerXYZ.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

#### 11.13.5.6 char const\* usage

**Initial value:**

```
"usage: demo-GetFingerXYZ [options]\n"
```

## 11.14 demo/demo-GetFingerXYZ.cpp File Reference

### 11.14.1 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.

### 11.14.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-03-07

#### Bug

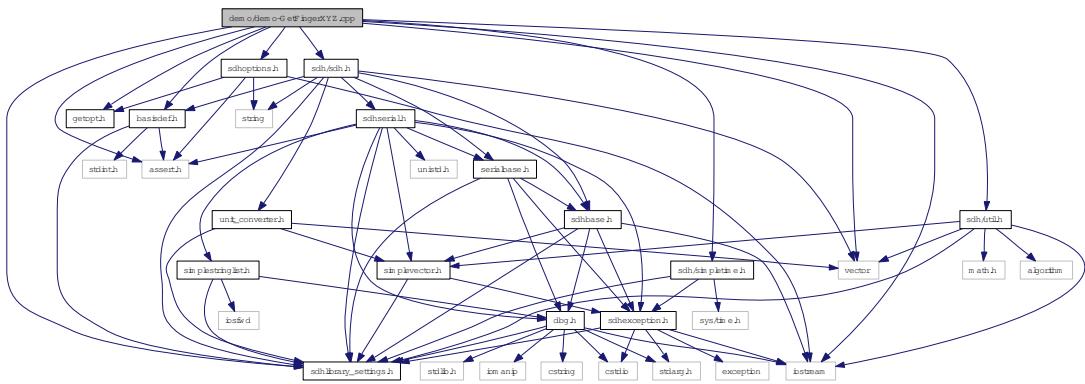
When compiled with MS Visual Studio then using the "-R" command line parameter will make the demonstration program demo-GetFingerXYZ abort.

### 11.14.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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`

### Some informative variables

- char const \* `_help_`
- char const \* `_author_` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `_url_` = "http://www.schunk.com"
- char const \* `_version_` = "\$Id: demo-GetFingerXYZ.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `_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_`

#### Initial value:

```
"Print measured XYZ position of fingertips of SDH.\n"
"C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"For every finger the actual axis angles and the finger tip coordinates\n"
"are printed.\n"
```

```
"\n"
"- Example usage:\n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to port 2 = COM3 once:\n"
"-       > demo-GetFingerXYZ -p 2\n"
"-     \n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to port 2 = COM3 every 500ms:\n"
"-       > demo-GetFingerXYZ -p 2 -t 0.5\n"
"-     \n"
"-   - Print finger angles and finger tip xyz coordinates of an SDH connected\n"
"-     to USB to RS232 converter 0 once:\n"
"-       > demo-GetFingerXYZ --sdh_rs_device=/dev/ttyUSB0 \n"
"-     \n"
"-   - Get the version info of an SDH connected to port 2 = COM3 \n"
"-       > demo-GetFingerXYZ --port=2 -v\n"
"- Known bugs:\n"
"-   - Command line parameter \"-R\" does not work when compiled \n"
"-     with MS Visual Studio\n"
```

#### 11.14.5.4 char const\* \_\_url\_\_ = "http://www.schunk.com"

#### 11.14.5.5 char const\* \_\_version\_\_ = "\$Id: demo-GetFingerXYZ.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

#### 11.14.5.6 char const\* usage

**Initial value:**

```
"usage: demo-GetFingerXYZ [options]\n"
```

## 11.15 `cpp.desire.final/demo/demo-mimic.cpp` File Reference

### **11.15.1 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.

### **11.15.2 General file information**

**Author:**

Dirk Osswald

Date:

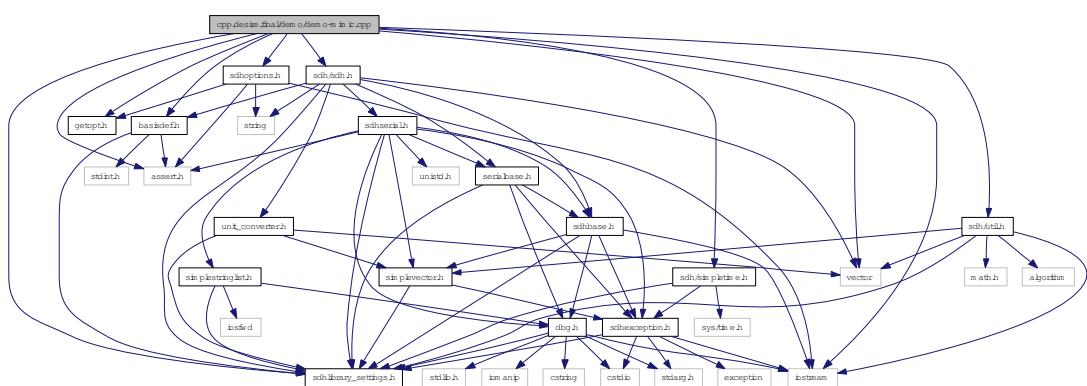
2007-03-07

### 11.15.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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 (cSDHOOptions &options, cDBG &cdbg, int nb)`
- `int main (int argc, char **argv)`

## Variables

### Some informative variables

- `char const * __help__`
- `char const * __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`
- `char const * __url__ = "http://www.schunk.com"`
- `char const * __version__ = "$Id: demo-mimic.cpp 4932 2009-11-02 08:20:24Z Osswald2 $"`
- `char const * __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

## 11.15.4 Function Documentation

### 11.15.4.1 `cSDH* GetHand (cSDHOOptions & options, cDBG & cdbg, int nb)`

### 11.15.4.2 `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__`

#### Initial value:

```
"In case you have 2 SDHs you can operate one of them by moving the first\n"
"hand manually. You must give parameters for 2 hands on the command line.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- Print actual angles of an SDH connected to port 2 = COM3 once:\n"
"> demo-mimic -p 2\n"
"\n"
"- Print actual angles of an SDH connected to port 2 = COM3 every 500ms:\n"
"> demo-mimic -p 2 -t 0.5\n"
"\n"
"- Print actual angles of an SDH connected to USB to RS232 converter 0 once:\n"
"> demo-mimic --sdh_rs_device=/dev/ttyUSB0\n"
"\n"
"- Get the version info of an SDH connected to port 2 = COM3 \n"
"> demo-mimic --port=2 -v\n"
```

### 11.15.5.4 `char const* __url__ = "http://www.schunk.com"`

### 11.15.5.5 `char const* __version__ = "$Id: demo-mimic.cpp 4932 2009-11-02 08:20:24Z Osswald2 $"`

## 11.16 demo/demo-mimic.cpp File Reference

### 11.16.1 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.

### 11.16.2 General file information

**Author:**

Dirk Osswald

**Date:**

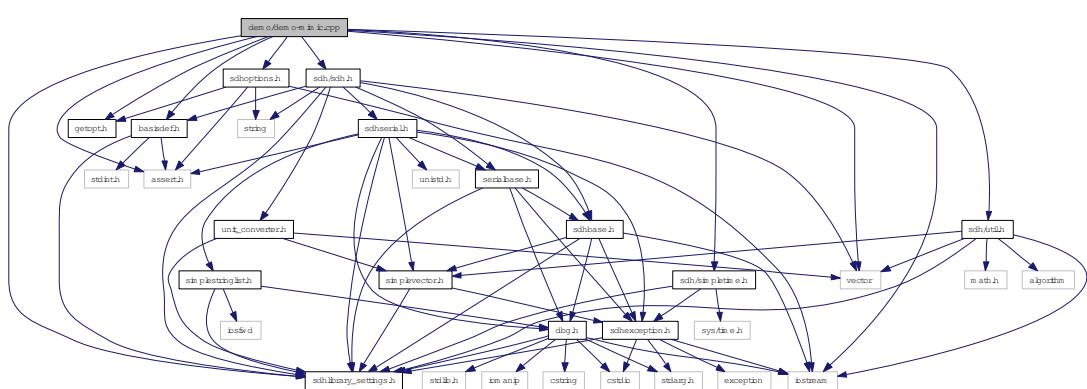
2007-03-07

### 11.16.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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__`
- `char const * __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"`
- `char const * __url__ = "http://www.schunk.com"`
- `char const * __version__ = "$Id: demo-mimic.cpp 5000 2009-12-01 10:41:18Z Osswald2 $"`
- `char const * __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"`

## 11.16.4 Function Documentation

### 11.16.4.1 `cSDH* GetHand (cSDHOptions & options, cDBG & cdbg, int nb)`

### 11.16.4.2 `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__`

#### Initial value:

```
"In case you have 2 SDHs you can operate one of them by moving the first\n"
"hand manually. You must give parameters for 2 hands on the command line.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- Print actual angles of an SDH connected to port 2 = COM3 once:\n"
"> demo-mimic -p 2\n"
"\n"
"- Print actual angles of an SDH connected to port 2 = COM3 every 500ms:\n"
"> demo-mimic -p 2 -t 0.5\n"
"\n"
"- Print actual angles of an SDH connected to USB to RS232 converter 0 once:\n"
"> demo-mimic --sdh_rs_device=/dev/ttyUSB0\n"
"\n"
"- Get the version info of an SDH connected to port 2 = COM3\n"
"> demo-mimic --port=2 -v\n"
```

### 11.16.5.4 `char const* __url__ = "http://www.schunk.com"`

### 11.16.5.5 `char const* __version__ = "$Id: demo-mimic.cpp 5000 2009-12-01 10:41:18Z Osswald2 $"`

## 11.17 cpp.desire.final/demo/demo-simple-withtiming.cpp File Reference

### 11.17.1 Detailed Description

Very simple C++ programm to make an attached SDH move.

### 11.17.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

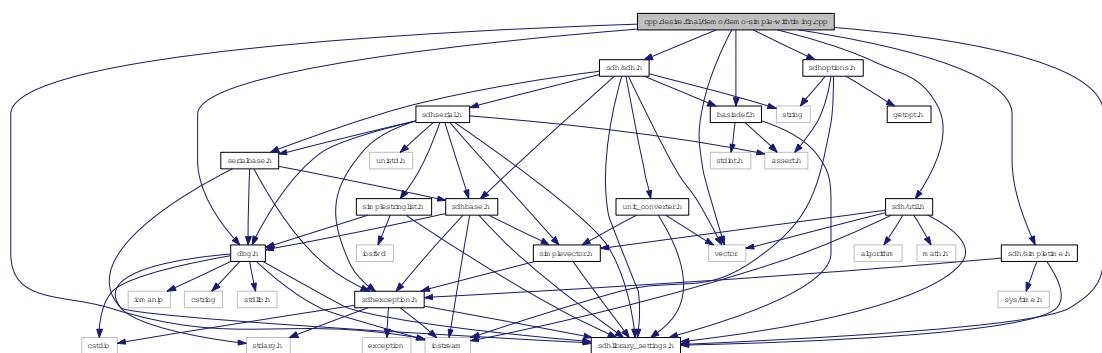
This code contains only the very basicst use of the features provided by the SDHLIBRARY-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.17.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### 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 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **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** = "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.)"

11.17.5.4 char const\* **url** = "http://www.schunk.com"

11.17.5.5 char const\* **version** = "\$Id: demo-simple-withtiming.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.17.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-simple-withtiming [options]\n"
```

## 11.18 demo/demo-simple-withtiming.cpp File Reference

### 11.18.1 Detailed Description

Very simple C++ programm to make an attached SDH move.

### 11.18.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

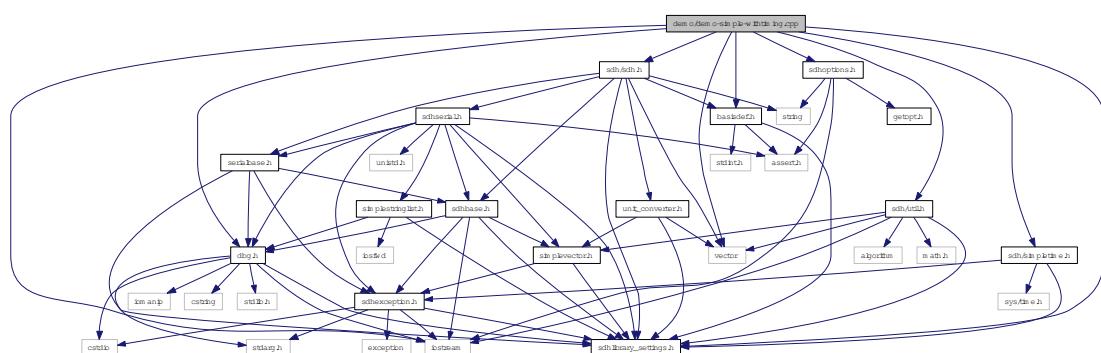
This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.18.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### 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 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **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** = "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.)"

11.18.5.4 char const\* **url** = "http://www.schunk.com"

11.18.5.5 char const\* **version** = "\$Id: demo-simple-withtiming.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.18.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-simple-withtiming [options]\n"
```

## 11.19 cpp.desire.final/demo/demo-simple.cpp File Reference

### 11.19.1 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.

### 11.19.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

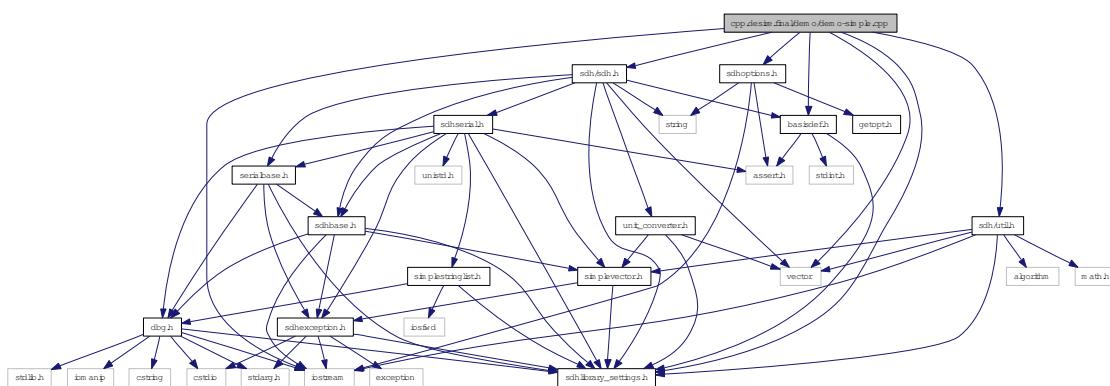
This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.19.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### Some informative variables

- char const \* **\_help\_**
- char const \* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **\_url\_** = "http://www.schunk.com"
- char const \* **\_version\_** = "\$Id: demo-simple.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.19.4 Function Documentation

### 11.19.4.1 int main (int *argc*, char \*\**argv*)

## 11.19.5 Variable Documentation

### 11.19.5.1 char const\* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.19.5.2 char const\* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.19.5.3 char const\* **\_help\_**

#### Initial value:

```
"Move proximal and distal joints of finger 1 three times by 10 degrees.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- Make SDH connected to port 2 = COM3 move:\n"
"> demo-simple -p 2\n"
"\n"
- Make SDH connected to USB to RS232 converter 0 move:\n"
"> demo-simple --sdh_rs_device=/dev/ttyUSB0\n"
"\n"
- Get the version info of an SDH connected to port 2 = COM3 \n"
"> demo-simple --port=2 -v\n"
```

### 11.19.5.4 char const\* **\_url\_** = "http://www.schunk.com"

### 11.19.5.5 char const\* **\_version\_** = "\$Id: demo-simple.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.19.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-simple [options]\n"
```

## 11.20 demo/demo-simple.cpp File Reference

### 11.20.1 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.

### 11.20.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

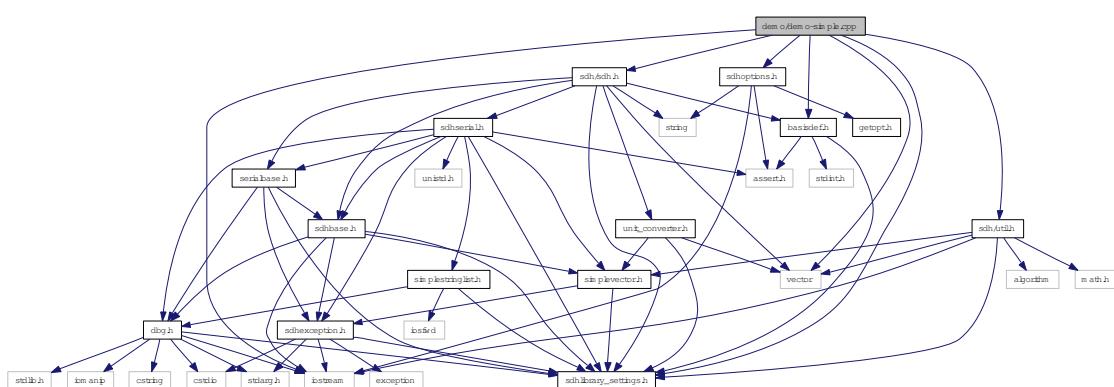
This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.20.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**

### Some informative variables

- char const \* **\_help\_**
- char const \* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **\_url\_** = "http://www.schunk.com"
- char const \* **\_version\_** = "\$Id: demo-simple.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.20.4 Function Documentation

### 11.20.4.1 int main (int *argc*, char \*\**argv*)

## 11.20.5 Variable Documentation

### 11.20.5.1 char const\* **\_author\_** = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.20.5.2 char const\* **\_copyright\_** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.20.5.3 char const\* **\_help\_**

#### Initial value:

```
"Move proximal and distal joints of finger 1 three times by 10 degrees.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- Make SDH connected to port 2 = COM3 move:\n"
"> demo-simple -p 2\n"
"\n"
"- Make SDH connected to USB to RS232 converter 0 move:\n"
"> demo-simple --sdh_rs_device=/dev/ttyUSB0\n"
"\n"
"- Get the version info of an SDH connected to port 2 = COM3 \n"
"> demo-simple --port=2 -v\n"
```

### 11.20.5.4 char const\* **\_url\_** = "http://www.schunk.com"

### 11.20.5.5 char const\* **\_version\_** = "\$Id: demo-simple.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

### 11.20.5.6 char const\* **usage**

#### Initial value:

```
"usage: demo-simple [options]\n"
```

## 11.21 cpp.desire.final/demo/demo-simple2.cpp File Reference

### 11.21.1 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.

### 11.21.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

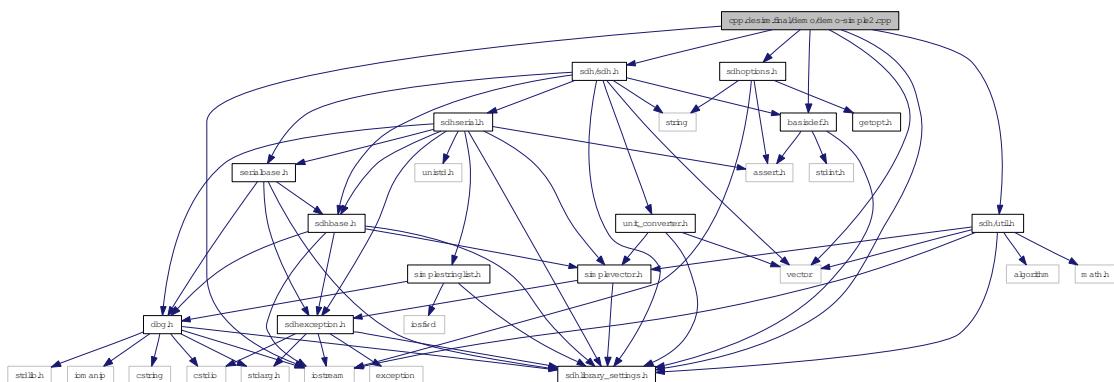
This code contains only the very basicst use of the features provided by the SDHLibrary-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.21.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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__`
- char const \* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `__url__` = "http://www.schunk.com"
- char const \* `__version__` = "\$Id: demo-simple2.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.21.4 Function Documentation

### 11.21.4.1 int main (int *argc*, char \*\* *argv*)

## 11.21.5 Variable Documentation

### 11.21.5.1 char const\* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.21.5.2 char const\* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.21.5.3 char const\* `__help__`

#### Initial value:

```
"Move proximal and distal joints of finger 1 three times by 10 degrees, stop movement when halfway done\n"
"- Example usage:\n"
"-   - Make SDH connected to port 2 = COM3 move:\n"
"-     > demo-simple2 -p 2\n"
"-     \n"
"-   - Make SDH connected to USB to RS232 converter 0 move:\n"
"-     > demo-simple2 --sdh_rs_device=/dev/ttyUSB0 \n"
"-     \n"
"-   - Get the version info of an SDH connected to port 2 = COM3 \n"
"-     > demo-simple2 --port=2 -v\n"
```

### 11.21.5.4 char const\* `__url__` = "http://www.schunk.com"

### 11.21.5.5 char const\* `__version__` = "\$Id: demo-simple2.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.22 demo/demo-simple2.cpp File Reference

### 11.22.1 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.

### 11.22.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

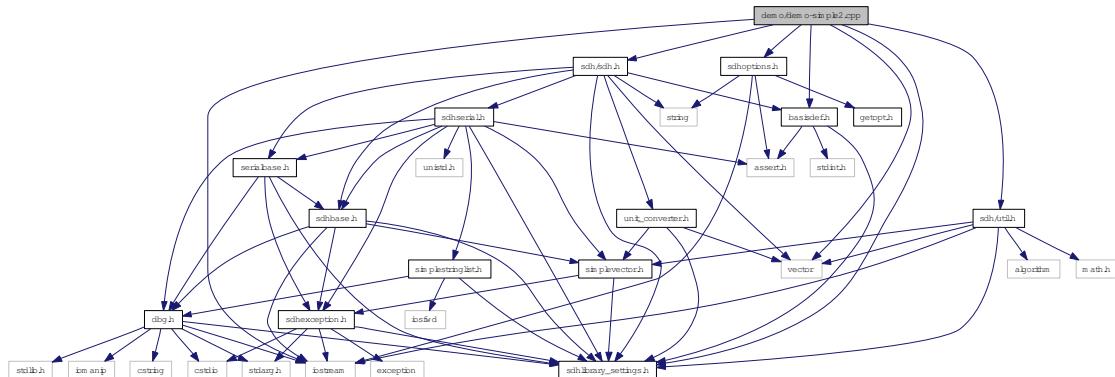
This code contains only the very basicst use of the features provided by the SDHLIBRARY-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.22.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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__`
- char const \* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `__url__` = "http://www.schunk.com"
- char const \* `__version__` = "\$Id: demo-simple2.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.22.4 Function Documentation

### 11.22.4.1 int main (int *argc*, char \*\* *argv*)

## 11.22.5 Variable Documentation

### 11.22.5.1 char const\* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.22.5.2 char const\* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.22.5.3 char const\* `__help__`

#### Initial value:

```
"Move proximal and distal joints of finger 1 three times by 10 degrees, stop movement when halfway done\n"
"- Example usage:\n"
"-   - Make SDH connected to port 2 = COM3 move:\n"
"-     > demo-simple2 -p 2\n"
"-     \n"
"-   - Make SDH connected to USB to RS232 converter 0 move:\n"
"-     > demo-simple2 --sdh_rs_device=/dev/ttyUSB0 \n"
"-     \n"
"-   - Get the version info of an SDH connected to port 2 = COM3 \n"
"-     > demo-simple2 --port=2 -v\n"
```

### 11.22.5.4 char const\* `__url__` = "http://www.schunk.com"

### 11.22.5.5 char const\* `__version__` = "\$Id: demo-simple2.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.23 cpp.desire.final/demo/demo-simple3.cpp File Reference

### 11.23.1 Detailed Description

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Wait-Axis. See [\\_help\\_](#) and online help (" -h " or " -help ") for available options.

### 11.23.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

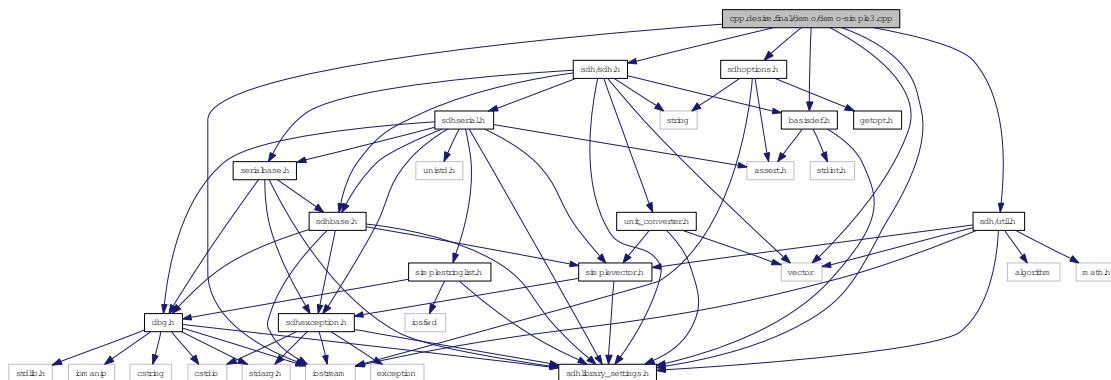
This code contains only the very basicst use of the features provided by the SDHLIBRARY-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.23.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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__`
- char const \* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `__url__` = "http://www.schunk.com"
- char const \* `__version__` = "\$Id: demo-simple3.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.23.4 Function Documentation

### 11.23.4.1 int main (int *argc*, char \*\* *argv*)

## 11.23.5 Variable Documentation

### 11.23.5.1 char const\* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.23.5.2 char const\* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.23.5.3 char const\* `__help__`

#### Initial value:

```
"Move axes 1,2 and 3 to a specific point.\n(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- - Make SDH connected to port 2 = COM3 move:\n"
"-   > demo-simple3 -p 2\n"
"- \n"
"- - Make SDH connected to USB to RS232 converter 0 move:\n"
"-   > demo-simple3 --sdh_rs_device=/dev/ttyUSB0 \n"
"- \n"
"- - Get the version info of an SDH connected to port 2 = COM3 \n"
"-   > demo-simple3 --port=2 -v\n"
```

### 11.23.5.4 char const\* `__url__` = "http://www.schunk.com"

### 11.23.5.5 char const\* `__version__` = "\$Id: demo-simple3.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.24 demo/demo-simple3.cpp File Reference

### 11.24.1 Detailed Description

Very simple C++ programm to make an attached SDH move. With non-sequential call of move and Wait-Axis. See [\\_help\\_](#) and online help (" -h " or " -help ") for available options.

### 11.24.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-01-18

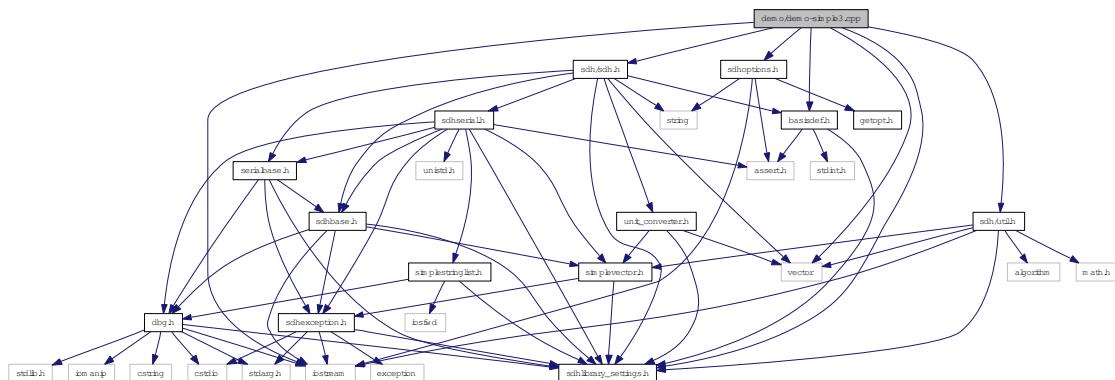
This code contains only the very basicst use of the features provided by the SDHLIBRARY-CPP. For more sophisticated applications see the other demo-\*.cpp programms, or of course the html/pdf documentation.

### 11.24.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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__`
- char const \* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `__url__` = "http://www.schunk.com"
- char const \* `__version__` = "\$Id: demo-simple3.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.24.4 Function Documentation

### 11.24.4.1 int main (int *argc*, char \*\* *argv*)

## 11.24.5 Variable Documentation

### 11.24.5.1 char const\* `__author__` = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.24.5.2 char const\* `__copyright__` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.24.5.3 char const\* `__help__`

#### Initial value:

```
"Move axes 1,2 and 3 to a specific point.\n(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- - Make SDH connected to port 2 = COM3 move:\n"
"-   > demo-simple3 -p 2\n"
"- \n"
"- - Make SDH connected to USB to RS232 converter 0 move:\n"
"-   > demo-simple3 --sdh_rs_device=/dev/ttyUSB0 \n"
"- \n"
"- - Get the version info of an SDH connected to port 2 = COM3 \n"
"-   > demo-simple3 --port=2 -v\n"
```

### 11.24.5.4 char const\* `__url__` = "http://www.schunk.com"

### 11.24.5.5 char const\* `__version__` = "\$Id: demo-simple3.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.25 cpp.desire.final/demo/demo-temperature.cpp File Reference

### 11.25.1 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.

### 11.25.2 General file information

#### Author:

Dirk Osswald

#### Date:

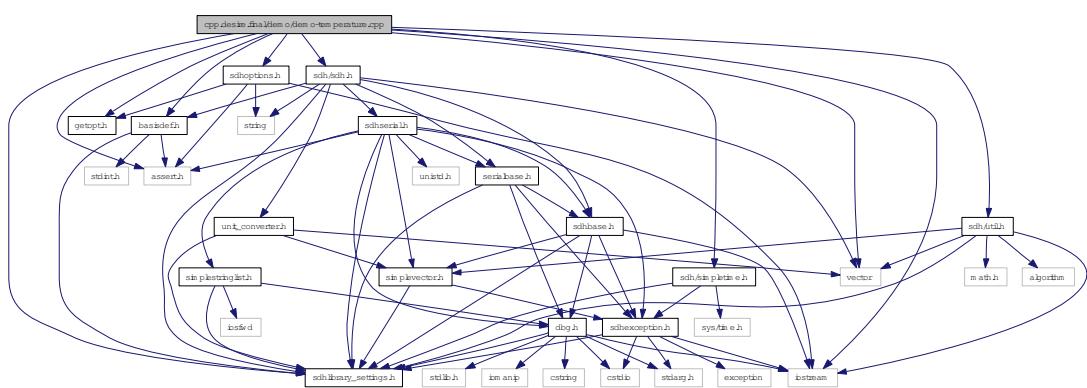
2007-01-18

### 11.25.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**
- char const \* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **url** = "http://www.schunk.com"
- char const \* **version** = "\$Id: demo-temperature.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **copyright** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.25.4 Function Documentation

### 11.25.4.1 int main (int *argc*, char \*\**argv*)

## 11.25.5 Variable Documentation

### 11.25.5.1 char const\* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.25.5.2 char const\* **copyright** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.25.5.3 char const\* **help**

**Initial value:**

```
"Print measured temperatures of SDH.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"A vector of temperatures is reported. The first 7 temperatures\n"
"are from sensors close to the corresponding axes motors.\n"
"The 8th value is the temperature of the FPGA, the controller chip (CPU).\n"
"The 9th value is the temperature of the PCB (Printed circuit board)\n"
"in the body of the SDH.\n"
"\n"
"- Example usage:\n"
" - Print temperatures of an SDH connected to port 2 = COM3 once:\n"
"   > demo-temperature -p 2\n"
"\n"
" - Print temperatures of an SDH connected to port 2 = COM3 every 500ms:\n"
"   > demo-temperature -p 2 -t 0.5\n"
"\n"
"- Print temperatures of an SDH connected to USB to RS232 converter 0 move:\n"
"   > demo-temperature --sdh_rs_device=/dev/ttyUSB0 \n"
"\n"
"- Get the version info of an SDH connected to port 2 = COM3 \n"
"   > demo-temperature --port=2 -v\n"
```

### 11.25.5.4 char const\* **url** = "http://www.schunk.com"

### 11.25.5.5 char const\* **version** = "\$Id: demo-temperature.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.26 demo/demo-temperature.cpp File Reference

### 11.26.1 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.

### 11.26.2 General file information

#### Author:

Dirk Osswald

#### Date:

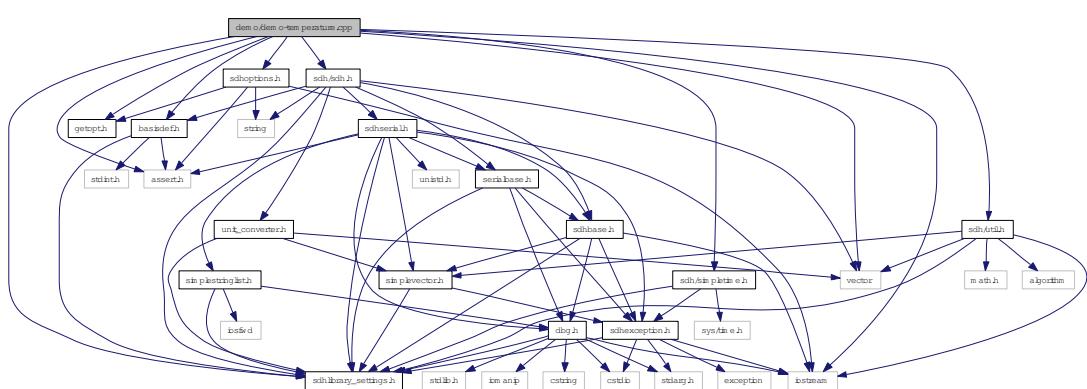
2007-01-18

### 11.26.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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**
- char const \* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* **url** = "http://www.schunk.com"
- char const \* **version** = "\$Id: demo-temperature.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* **copyright** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.26.4 Function Documentation

### 11.26.4.1 int main (int *argc*, char \*\**argv*)

## 11.26.5 Variable Documentation

### 11.26.5.1 char const\* **author** = "Dirk Osswald: dirk.osswald@de.schunk.com"

### 11.26.5.2 char const\* **copyright** = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

### 11.26.5.3 char const\* **help**

**Initial value:**

```
"Print measured temperatures of SDH.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"A vector of temperatures is reported. The first 7 temperatures\n"
"are from sensors close to the corresponding axes motors.\n"
"The 8th value is the temperature of the FPGA, the controller chip (CPU).\n"
"The 9th value is the temperature of the PCB (Printed circuit board)\n"
"in the body of the SDH.\n"
"\n"
"- Example usage:\n"
" - Print temperatures of an SDH connected to port 2 = COM3 once:\n"
"   > demo-temperature -p 2\n"
"\n"
" - Print temperatures of an SDH connected to port 2 = COM3 every 500ms:\n"
"   > demo-temperature -p 2 -t 0.5\n"
"\n"
"- Print temperatures of an SDH connected to USB to RS232 converter 0 move:\n"
"   > demo-temperature --sdh_rs_device=/dev/ttyUSB0 \n"
"\n"
"- Get the version info of an SDH connected to port 2 = COM3 \n"
"   > demo-temperature --port=2 -v\n"
```

### 11.26.5.4 char const\* **url** = "http://www.schunk.com"

### 11.26.5.5 char const\* **version** = "\$Id: demo-temperature.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"

## 11.27 cpp.desire.final/demo/demo-velocity-acceleration.cpp File Reference

### 11.27.1 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.

### 11.27.2 General file information

#### Author:

Dirk Osswald

#### Date:

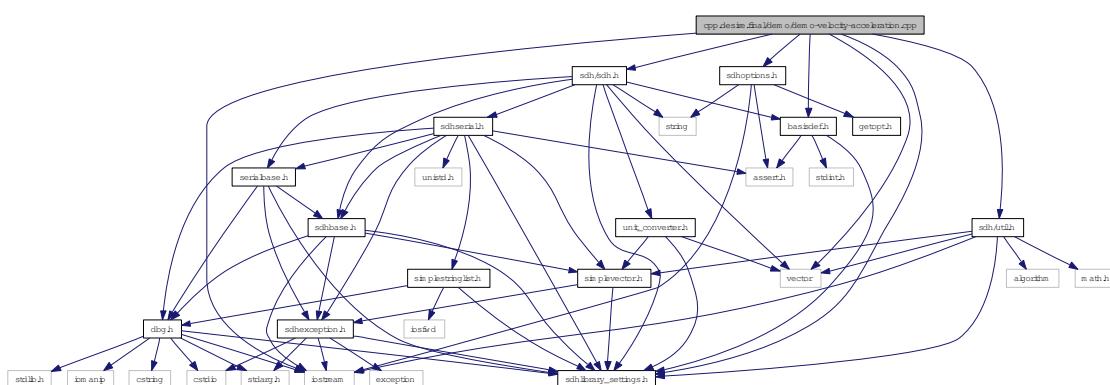
2007-01-18

### 11.27.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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-velocity-acceleration.cpp:



### Functions

- int [main](#) (int argc, char \*\*argv)

## Variables

- char const \* `usage`

### Some informative variables

- char const \* `_help_`
- 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 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.27.4 Function Documentation

### 11.27.4.1 int main (int *argc*, char \*\* *argv*)

## 11.27.5 Variable Documentation

**11.27.5.1 char const\* `_author_` = "Dirk Osswald: dirk.osswald@de.schunk.com"**

**11.27.5.2 char const\* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"**

**11.27.5.3 char const\* `_help_`**

**Initial value:**

```
"Make the SDH move one finger in \"velocity with acceleration ramp\" control mode.\n(C++ demo applicat\n"
"\n"
"- Example usage:\n"
" - Make SDH connected to port 2 = COM3 move:\n"
"   > demo-velocity-acceleration -p 2\n"
"   \n"
" - Make SDH connected to USB to RS232 converter 0 move:\n"
"   > demo-velocity-acceleration --sdh_rs_device=/dev/ttyUSBO \n"
"   \n"
" - Get the version info of an SDH connected to port 2 = COM3 \n"
"   > demo-velocity-acceleration --port=2 -v\n"
```

**11.27.5.4 char const\* `_url_` = "http://www.schunk.com"**

**11.27.5.5 char const\* `_version_` = "\$Id: demo-velocity-acceleration.cpp 4932 2009-11-02\n08:20:24Z Osswald2 \$"**

**11.27.5.6 char const\* `usage`**

**Initial value:**

```
"usage: demo-velocity-acceleration [options]\n"
```

## 11.28 demo/demo-velocity-acceleration.cpp File Reference

### 11.28.1 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.

### 11.28.2 General file information

#### Author:

Dirk Osswald

#### Date:

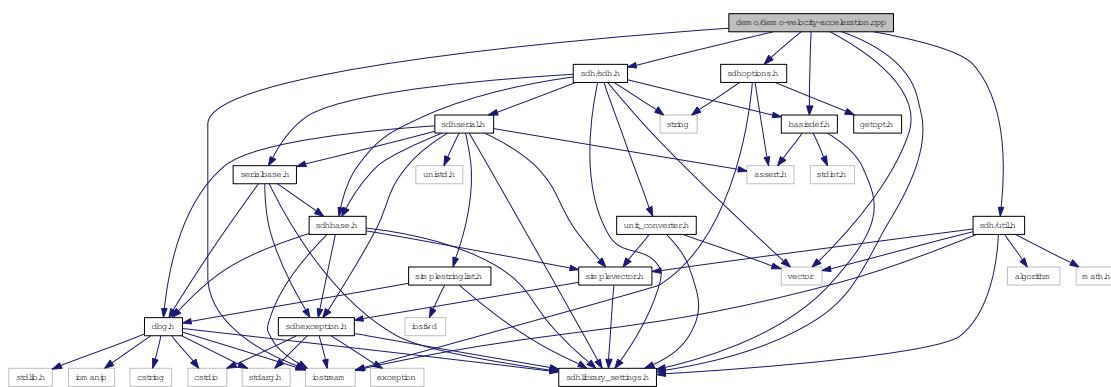
2007-01-18

### 11.28.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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-velocity-acceleration.cpp:



## Functions

- int [main](#) (int argc, char \*\*argv)

## Variables

- char const \* `usage`

### Some informative variables

- char const \* `_help_`
- 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 4932 2009-11-02 08:20:24Z Osswald2 \$"
- char const \* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.28.4 Function Documentation

### 11.28.4.1 int main (int *argc*, char \*\* *argv*)

## 11.28.5 Variable Documentation

**11.28.5.1 char const\* `_author_` = "Dirk Osswald: dirk.osswald@de.schunk.com"**

**11.28.5.2 char const\* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"**

**11.28.5.3 char const\* `_help_`**

**Initial value:**

```
"Make the SDH move one finger in \"velocity with acceleration ramp\" control mode.\n(C++ demo applicat\n"\n"- Example usage:\n"- - Make SDH connected to port 2 = COM3 move:\n" -> demo-velocity-acceleration -p 2\n"- \n"- - Make SDH connected to USB to RS232 converter 0 move:\n" -> demo-velocity-acceleration --sdh_rs_device=/dev/ttyUSBO \n"- \n"- - Get the version info of an SDH connected to port 2 = COM3 \n"- > demo-velocity-acceleration --port=2 -v\n"
```

**11.28.5.4 char const\* `_url_` = "http://www.schunk.com"**

**11.28.5.5 char const\* `_version_` = "\$Id: demo-velocity-acceleration.cpp 4932 2009-11-02 08:20:24Z Osswald2 \$"**

**11.28.5.6 char const\* `usage`**

**Initial value:**

```
"usage: demo-velocity-acceleration [options]\n"
```

## 11.29 cpp.desire.final/demo/dsaboost.cpp File Reference

### 11.29.1 Detailed Description

helper stuff for the "boosted" DSA stuff

### 11.29.2 General file information

**Author:**

Dirk Osswald

**Date:**

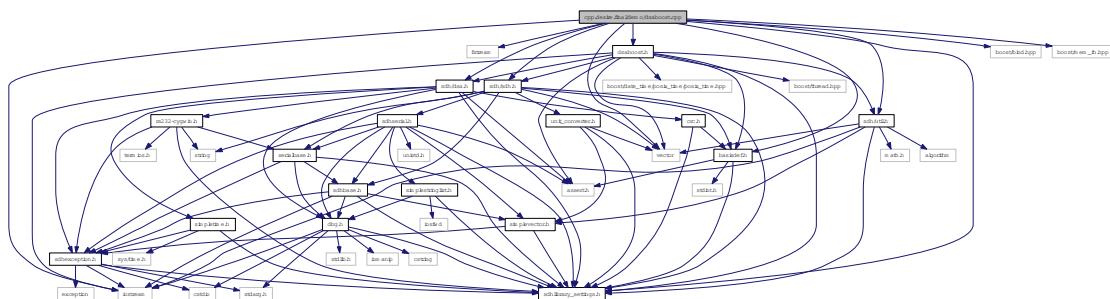
2009-08-02

### 11.29.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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

- cDBG [cdbg](#)

## 11.29.4 Variable Documentation

### 11.29.4.1 cDBG cdbg

## 11.30 demo/dsabooost.cpp File Reference

### 11.30.1 Detailed Description

helper stuff for the "boosted" DSA stuff

### 11.30.2 General file information

#### Author:

Dirk Osswald

#### Date:

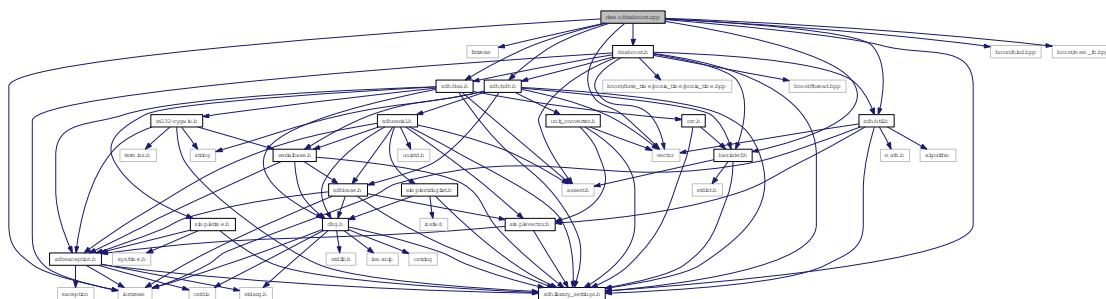
2009-08-02

### 11.30.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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 "dsabooost.h"
#include <boost/bind.hpp>
#include <boost/mem_fn.hpp>
```

Include dependency graph for dsabooost.cpp:



### Variables

- cDBG [cdbg](#)

## 11.30.4 Variable Documentation

### 11.30.4.1 cDBG cdbg

## 11.31 cpp.desire.final/demo/dsaboost.h File Reference

### 11.31.1 Detailed Description

helper stuff for the DSA using boost

### 11.31.2 General file information

**Author:**

Dirk Osswald

**Date:**

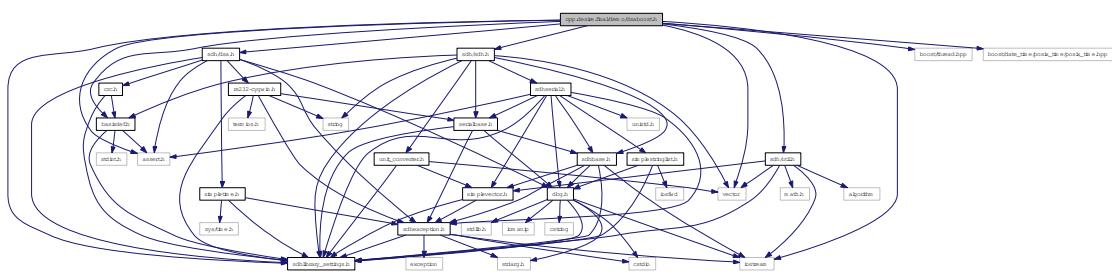
2009-08-03

### 11.31.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cDSAUpdater](#)
- class [SDH::cIsGraspedBase](#)
- class [SDH::cIsGraspedByArea](#)

## 11.32 demo/dsaboostr.h File Reference

### 11.32.1 Detailed Description

helper stuff for the DSA using boost

### **11.32.2 General file information**

**Author:**

Dirk Osswald

Date:

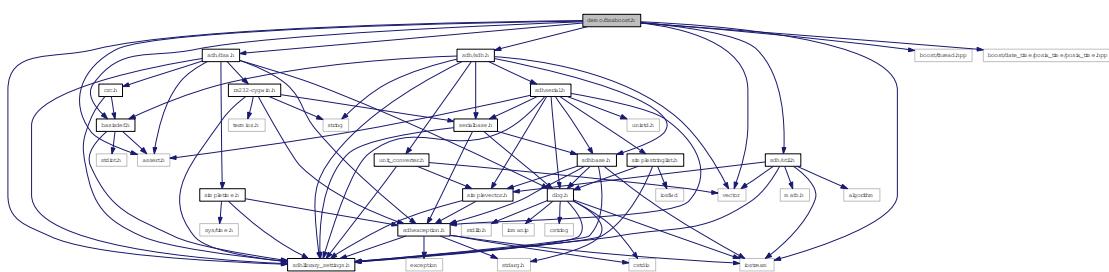
2009-08-03

### 11.32.3 Copyright

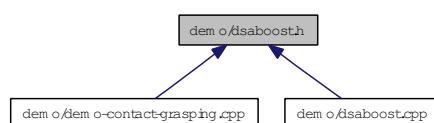
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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.hpp"
```

Include dependency graph for dsaboostr.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cDSAUpdater](#)
- class [SDH::cIsGraspedBase](#)
- class [SDH::cIsGraspedByArea](#)

## 11.33 cpp.desire.final/demo/sdhoptions.cpp File Reference

### 11.33.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

**Author:**

Dirk Osswald

Date:

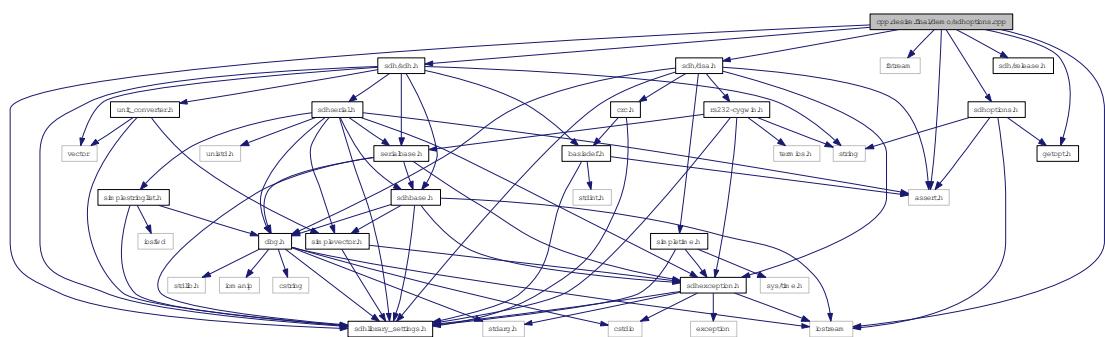
2008-05-05

### 11.33.2 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <fstream>
#include "sdh/sdh.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/release.h"
#include "sdh/dsa.h"
#include "sdhoptions.h"
```

## Include dependency graph for sdhoptions.cpp:



## Variables

- static char const \* sdhusage\_general  
*general options*
  - static char const \* sdhusage\_sdhcom\_serial  
*RS232 communication options.*

- static char const \* [sdhusage\\_sdhcom\\_common](#)  
*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\\_sdhoother](#)  
*Other options.*
- static char const \* [sdhusage\\_dsaicom](#)  
*DSA (tactile sensor) communication options.*
- static char const \* [sdhusage\\_dsaother](#)  
*DSA (tactile sensor) other options.*
- static char const \* [sdhoptions\\_short\\_options](#) = "hvd:l:p:T:b:cn:e:w:RFt:q:r:fSCM"
- static struct [option sdhoptions\\_long\\_options](#) [ ]

### 11.33.3 Variable Documentation

#### 11.33.3.1 struct option sdhoptions\_long\_options[] [static]

**Initial value:**

```
{
    {"help" , 0 , 0 , 'h' },
    {"version" , 0 , 0 , 'v' },
    {"debug" , 1 , 0 , 'd' },
    {"debuglog" , 1 , 0 , 'l' },

    {"port" , 1 , 0 , 'p' },
    {"sdhport" , 1 , 0 , 'p' },
    {"sdh_rs_device" , 1 , 0 , 'S' + 256},
    {"timeout" , 1 , 0 , 'T' },
    {"baud" , 1 , 0 , 'b' },

    {"can" , 0 , 0 , 'c' },
    {"canesd" , 0 , 0 , 'c' },
    {"net" , 1 , 0 , 'n' },

    {"canpeak" , 0 , 0 , 'p' + 256},
    {"sdh_canpeak_device" , 1 , 0 , 'P' + 256},

    {"id_read" , 1 , 0 , 'e' },
    {"id_write" , 1 , 0 , 'w' },

    {"radians" , 0 , 0 , 'R' },
    {"fahrenheit" , 0 , 0 , 'F' }
}
```

```

    {"period"      , 1      , 0      , 't'      },
    {"dsaport"     , 1      , 0      , 'q'      },
    {"dsa_rs_device" , 1      , 0      , 'D' + 256},
    {"no_rle"      , 0      , 0      , 'r' + 256},
    {"framerate"   , 1      , 0      , 'r'      },
    {"fullframe"   , 0      , 0      , 'f'      },
    {"sensorinfo"  , 0      , 0      , 'S'      },
    {"controllerinfo", 0     , 0     , 'C'      },
    {"matrixinfo"  , 1      , 0      , 'M'      },
    {0, 0, 0, 0}
}

```

### 11.33.3.2 **char const\* sdhoptions\_short\_options = "hvdl:p:T:b:cn:e:w:RFt:q:r:fSCM"**

[static]

### 11.33.3.3 **char const\* sdhusage\_dsacom [static]**

#### Initial value:

```

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

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

" --no_rle\n"
"     Do not use the RunLengthEncoding\n"
" \n"

```

DSA (tactile sensor) communication options.

### 11.33.3.4 **char const\* sdhusage\_dsaother [static]**

#### Initial value:

```

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

" -f, --fullframe\n"
"     Print acquired full frames numerically.\n"
" \n"

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

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

```

```

" -M, --matrixinfo=MATRIX_INDEX\n"
"     Print matrix info for matrix with index MATRIX_INDEX from DSA.\n"
"     \n"

```

DSA (tactile sensor) other options.

### **11.33.3.5 char const\* sdhusage\_general [static]**

**Initial value:**

```

"General options:\n"
" -h, --help\n"
"     Show this help message and exit.\n"
"     \n"
" -v, --version\n"
"     Print the version (revision/release names) and dates of application,\n"
"     library (and the attached SDH firmware, if found), then exit.\n"
"     \n"
" -d LEVEL, --debug=LEVEL\n"
"     Print debug messages of level LEVEL or lower while executing the program.\n"
"     Level 0 (default): No messages, 1: application-level messages, \n"
"     2: CSDH-level messages, 3: CSDHSerial-level messages\n"
"     \n"
" -l LOGFILE, --debuglog=LOGFILE\n"
"     Redirect the printed debug messages to LOGFILE instead of default \n"
"     standard error. If LOGFILE starts with '+' then the output will be \n"
"     appended to the file (without the leading '+'), else the file will be\n"
"     overwritten.\n"
"     \n"

```

general options

### **11.33.3.6 char const\* sdhusage\_sdhcom\_cancommon [static]**

**Initial value:**

```
" "
```

Common CAN communication options.

### **11.33.3.7 char const\* sdhusage\_sdhcom\_common [static]**

**Initial value:**

```

" -T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from\n"
"     SDH. The default -1 means: wait forever.\n"
"     \n"
" -b BAUDRATE, --baud=BAUDRATE\n"
"     Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232\n"

```

Common communication options.

#### 11.33.3.8 char const\* sdhusage\_sdhcom\_esdcan [static]

**Initial value:**

""

ESD CAN communication options.

#### 11.33.3.9 char const\* sdhusage\_sdhcom\_peakcan [static]

**Initial value:**

""

PEAK CAN communication options.

#### 11.33.3.10 char const\* sdhusage\_sdhcom\_serial [static]

**Initial value:**

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

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

RS232 communication options.

#### 11.33.3.11 char const\* sdhusage\_sdhoother [static]

**Initial value:**

```
"Other options:\n"
"  -R, --radians\n"
"    Use radians and radians per second for angles and angular velocities\n"
"    instead of default degrees and degrees per second.\n"
"    \n"
"  -F, --fahrenheit\n"
"    Use degrees fahrenheit to report temperatures instead of default degrees\n"
"    celsius.\n"
"    \n"
"  -t PERIOD, --period=PERIOD\n"
"    For periodic commands only: Time period of measurements in seconds. The\n"
"    default of '0' means: report once only. If set then the time since start\n"
"    of measurement is printed at the beginning of every line.\n"
"    \n"
```

Other options.

## 11.34 demo/sdhoptions.cpp File Reference

### **11.34.1 Detailed Description**

Implementation of a class to parse common SDH related command line options.

**Author:**

Dirk Osswald

Date:

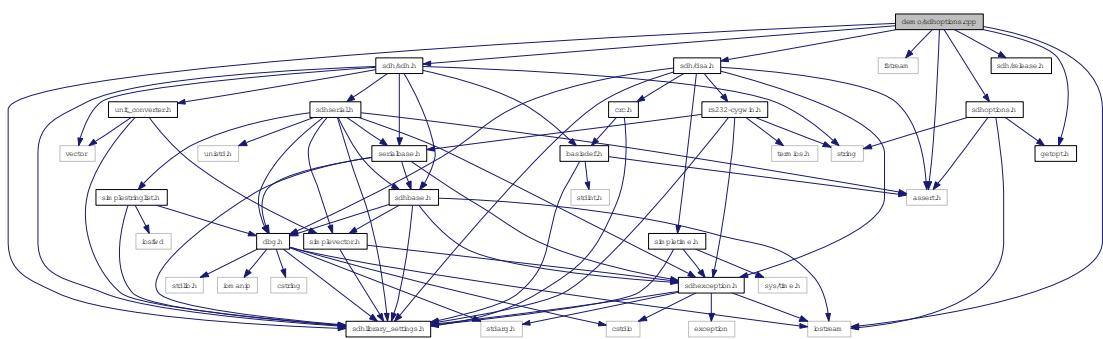
2008-05-05

### 11.34.2 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <fstream>
#include "sdh/sdh.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/release.h"
#include "sdh/dsa.h"
#include "sdhoptions.h"
```

Include dependency graph for sdhoptions.cpp:



## Variables

- static char const \* sdhusage\_general  
*general options*
  - static char const \* sdhusage\_sdhcom\_serial  
*RS232 communication options.*

- static char const \* [sdhusage\\_sdhcom\\_common](#)  
*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\\_sdhoother](#)  
*Other options.*
- static char const \* [sdhusage\\_dsaicom](#)  
*DSA (tactile sensor) communication options.*
- static char const \* [sdhusage\\_dsaother](#)  
*DSA (tactile sensor) other options.*
- static char const \* [sdhoptions\\_short\\_options](#) = "hvd:l:p:T:b:cn:e:w:RFt:q:r:fSCM:"
- static struct [option sdhoptions\\_long\\_options](#) [ ]

### 11.34.3 Variable Documentation

#### 11.34.3.1 struct option sdhoptions\_long\_options[] [static]

**Initial value:**

```
{
    {"help" , 0 , 0 , 'h' },
    {"version" , 0 , 0 , 'v' },
    {"debug" , 1 , 0 , 'd' },
    {"debuglog" , 1 , 0 , 'l' },

    {"port" , 1 , 0 , 'p' },
    {"sdhport" , 1 , 0 , 'p' },
    {"sdh_rs_device" , 1 , 0 , 'S' + 256},
    {"timeout" , 1 , 0 , 'T' },
    {"baud" , 1 , 0 , 'b' },

    {"can" , 0 , 0 , 'c' },
    {"canesd" , 0 , 0 , 'c' },
    {"net" , 1 , 0 , 'n' },

    {"canpeak" , 0 , 0 , 'p' + 256},
    {"sdh_canpeak_device" , 1 , 0 , 'P' + 256},

    {"id_read" , 1 , 0 , 'e' },
    {"id_write" , 1 , 0 , 'w' },

    {"radians" , 0 , 0 , 'R' },
    {"fahrenheit" , 0 , 0 , 'F' }
}
```

```

    {"period"      , 1      , 0      , 't'      },
    {"dsaport"     , 1      , 0      , 'q'      },
    {"dsa_rs_device" , 1      , 0      , 'D' + 256},
    {"no_rle"      , 0      , 0      , 'r' + 256},
    {"framerate"   , 1      , 0      , 'r'      },
    {"fullframe"   , 0      , 0      , 'f'      },
    {"sensorinfo"  , 0      , 0      , 'S'      },
    {"controllerinfo", 0     , 0     , 'C'      },
    {"matrixinfo"  , 1      , 0      , 'M'      },
    {0, 0, 0, 0}
}

```

**11.34.3.2 char const\* sdhoptions\_short\_options = "hvdl:p:T:b:cn:e:w:RFt:q:r:fSCM:"**  
[static]

**11.34.3.3 char const\* sdhusage\_dsacom [static]**

**Initial value:**

```

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

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

" --no_rle\n"
"     Do not use the RunLengthEncoding\n"
" \n"

```

DSA (tactile sensor) communication options.

**11.34.3.4 char const\* sdhusage\_dsaother [static]**

**Initial value:**

```

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

" -f, --fullframe\n"
"     Print acquired full frames numerically.\n"
" \n"

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

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

```

```

" -M, --matrixinfo=MATRIX_INDEX\n"
"     Print matrix info for matrix with index MATRIX_INDEX from DSA.\n"
"     \n"

```

DSA (tactile sensor) other options.

### **11.34.3.5 char const\* sdhusage\_general [static]**

**Initial value:**

```

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

```

general options

### **11.34.3.6 char const\* sdhusage\_sdhcom\_cancommon [static]**

**Initial value:**

```
" "
```

Common CAN communication options.

### **11.34.3.7 char const\* sdhusage\_sdhcom\_common [static]**

**Initial value:**

```

" -T TIMEOUT, --timeout=TIMEOUT Timeout in seconds when waiting for data from\n"
"     SDH. The default -1 means: wait forever.\n"
"     \n"
" -b BAUDRATE, --baud=BAUDRATE\n"
"     Use BAUDRATE in bit/s for communication. Default=115200 Bit/s for RS232\n"

```

Common communication options.

#### 11.34.3.8 `char const* sdhusage_sdhcom_esdcan [static]`

**Initial value:**

```
""
```

ESD CAN communication options.

#### 11.34.3.9 `char const* sdhusage_sdhcom_peakcan [static]`

**Initial value:**

```
""
```

PEAK CAN communication options.

#### 11.34.3.10 `char const* sdhusage_sdhcom_serial [static]`

**Initial value:**

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

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

RS232 communication options.

#### 11.34.3.11 `char const* sdhusage_sdhoother [static]`

**Initial value:**

```
"Other options:\n"
"  -R, --radians\n"
"    Use radians and radians per second for angles and angular velocities\n"
"    instead of default degrees and degrees per second.\n"
"    \n"
"  -F, --fahrenheit\n"
"    Use degrees fahrenheit to report temperatures instead of default degrees\n"
"    celsius.\n"
"    \n"
"  -t PERIOD, --period=PERIOD\n"
"    For periodic commands only: Time period of measurements in seconds. The\n"
"    default of '0' means: report once only. If set then the time since start\n"
"    of measurement is printed at the beginning of every line.\n"
"    \n"
```

Other options.

## 11.35 cpp.desire.final/demo/sdhoptions.h File Reference

### 11.35.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

### 11.35.2 General file information

#### Author:

Dirk Osswald

#### Date:

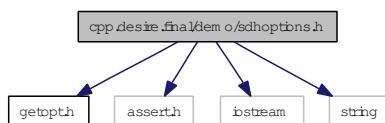
2008-05-05

### 11.35.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <string>
```

Include dependency graph for sdhoptions.h:



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



## Classes

- class [cSDHOptions](#)

## Defines

- #define [SDHUSAGE\\_DEFAULT](#) "general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon"
   
*string defining all the usage helptexts included by default*

## 11.35.4 Define Documentation

**11.35.4.1 #define SDHUSAGE\_DEFAULT "general sdhcom\_serial sdhcom\_common  
sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon"**

string defining all the usage helptexts included by default

## 11.36 demo/sdhoptions.h File Reference

### 11.36.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

### 11.36.2 General file information

#### Author:

Dirk Osswald

#### Date:

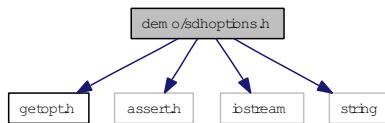
2008-05-05

### 11.36.3 Copyright

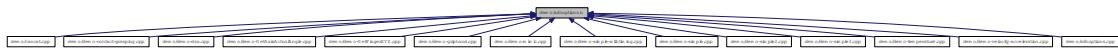
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <string>
```

Include dependency graph for sdhoptions.h:



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



## Classes

- class [cSDHOptions](#)

## Defines

- #define [SDHUSAGE\\_DEFAULT](#) "general sdhcom\_serial sdhcom\_common sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon"
   
*string defining all the usage helptexts included by default*

## 11.36.4 Define Documentation

**11.36.4.1 #define SDHUSAGE\_DEFAULT "general sdhcom\_serial sdhcom\_common  
sdhcom\_esdcan sdhcom\_peakcan sdhcom\_cancommon"**

string defining all the usage helptexts included by default

## 11.37 cpp.desire.final/sdh/basisdef.h File Reference

### 11.37.1 Detailed Description

This file contains some basic definitions (defines, macros, datatypes).

### 11.37.2 General file information

#### Author:

Jan Grewe, Dirk Osswald

#### Date:

08.10.2004

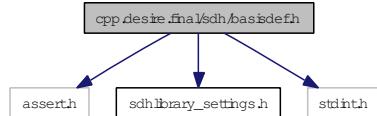
- Datatypes: SDH::Int8, SDH::UInt8, SDH::Int16, SDH::UInt16, SDH::Int32, SDH::UInt32

### 11.37.3 Copyright

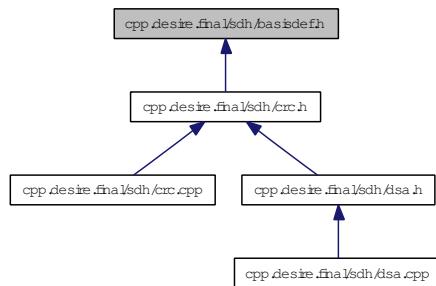
Copyright (c) 2006 SCHUNK GmbH & Co. KG

```
#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)*

## 11.37.4 Define Documentation

### 11.37.4.1 #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

## 11.38 `sdh/basisdef.h` File Reference

### 11.38.1 Detailed Description

This file contains some basic definitions (defines, macros, datatypes).

### 11.38.2 General file information

#### Author:

Jan Grewe, Dirk Osswald

#### Date:

08.10.2004

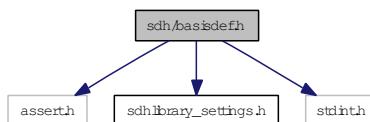
- Datatypes: SDH::Int8, SDH::UInt8, SDH::Int16, SDH::UInt16, SDH::Int32, SDH::UInt32

### 11.38.3 Copyright

Copyright (c) 2006 SCHUNK GmbH & Co. KG

```
#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*

## 11.38.4 Define Documentation

### 11.38.4.1 #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

## 11.39 cpp.desire.final/sdh/canserial-esd.cpp File Reference

### 11.39.1 Detailed Description

Implementation of class **SDH::cCANSerial\_ESD**, a class to access an ESD CAN interface on cygwin/linux and Visual Studio.

### 11.39.2 General file information

#### Author:

Dirk Osswald

#### Date:

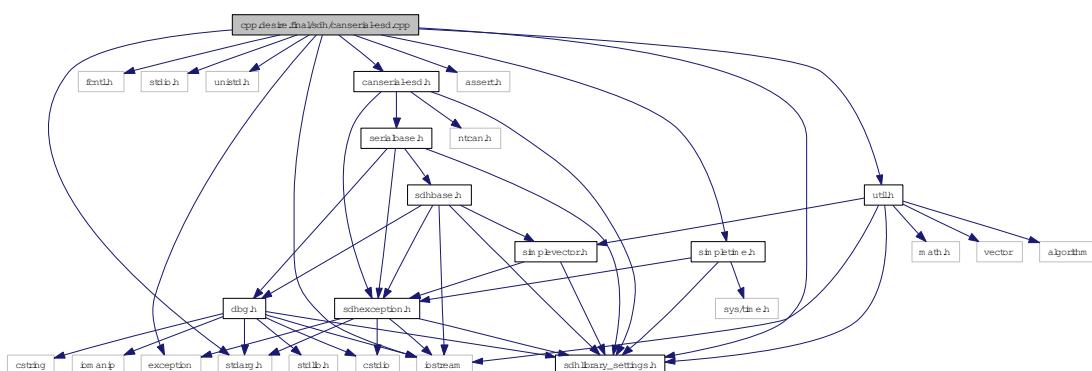
2007-02-20

### 11.39.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Functions

- char const \* [ESD\\_strerror](#) (NTCAN\_RESULT rc)

### 11.39.4 Function Documentation

#### 11.39.4.1 char const\* ESD\_strerror (NTCAN\_RESULT *rc*)

## 11.40 sdh/canserial-esd.cpp File Reference

### 11.40.1 Detailed Description

Implementation of class [SDH::cCANSerial\\_ESD](#), a class to access an ESD CAN interface on cygwin/linux and Visual Studio.

### 11.40.2 General file information

#### Author:

Dirk Osswald

#### Date:

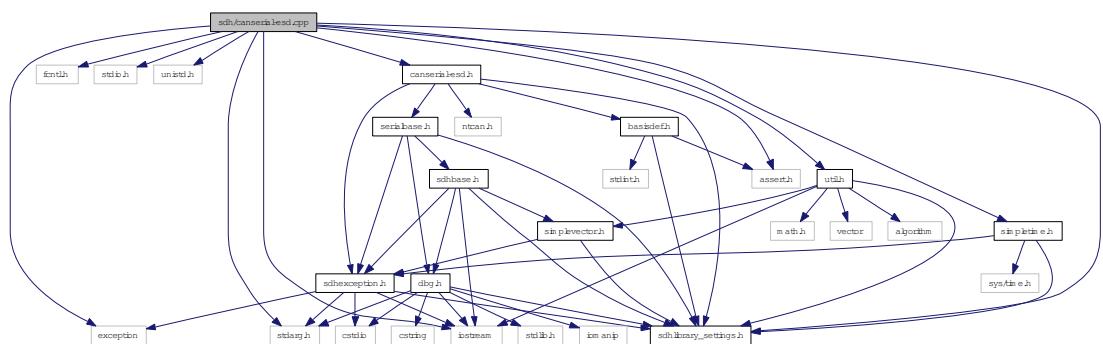
2007-02-20

### 11.40.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Functions

- `char const * ESD_strerror (NTCAN_RESULT rc)`

### 11.40.4 Function Documentation

#### 11.40.4.1 `char const* ESD_strerror (NTCAN_RESULT rc)`

## 11.41 cpp.desire.final/sdh/canserial-esd.h File Reference

### 11.41.1 Detailed Description

Interface of class `SDH::cCANSerial_ESD`, class to access CAN bus via ESD card on cygwin/linux.

### 11.41.2 General file information

#### Author:

Dirk Osswald

#### Date:

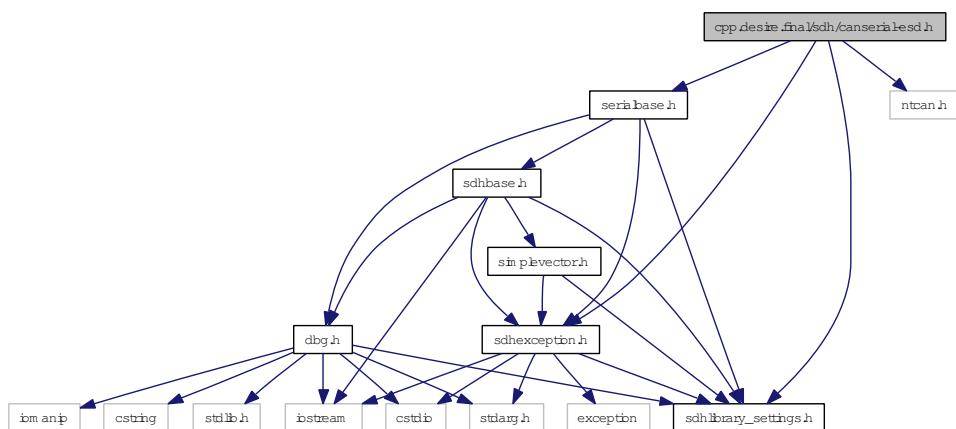
2008-05-02

### 11.41.3 Copyright

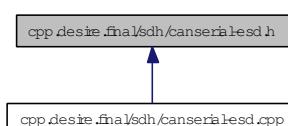
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhexception.h"
#include "serialbase.h"
#include "ntcan.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for canserial-esd.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cCANSerial\\_ESDEception](#)  
*Derived exception class for low-level CAN ESD related exceptions.*
- class [SDH::cCANSerial\\_ESD](#)  
*Low-level communication class to access a CAN port.*

## Defines

- #define [NOMINMAX](#)
- #define [CAN\\_ESD\\_TXQUEUESIZE](#) 32  
*transmit queue size for CAN frames*
- #define [CAN\\_ESD\\_RXQUEUESIZE](#) 512  
*receive queue size for CAN frames*

### 11.41.4 Define Documentation

#### 11.41.4.1 #define CAN\_ESD\_RXQUEUESIZE 512

receive queue size for CAN frames

#### 11.41.4.2 #define CAN\_ESD\_TXQUEUESIZE 32

transmit queue size for CAN frames

#### 11.41.4.3 #define NOMINMAX

## 11.42 sdh/canserial-esd.h File Reference

### 11.42.1 Detailed Description

Interface of class `SDH::cCANSerial_ESD`, class to access CAN bus via ESD card on cygwin/linux.

### 11.42.2 General file information

**Author:**

Dirk Osswald

**Date:**

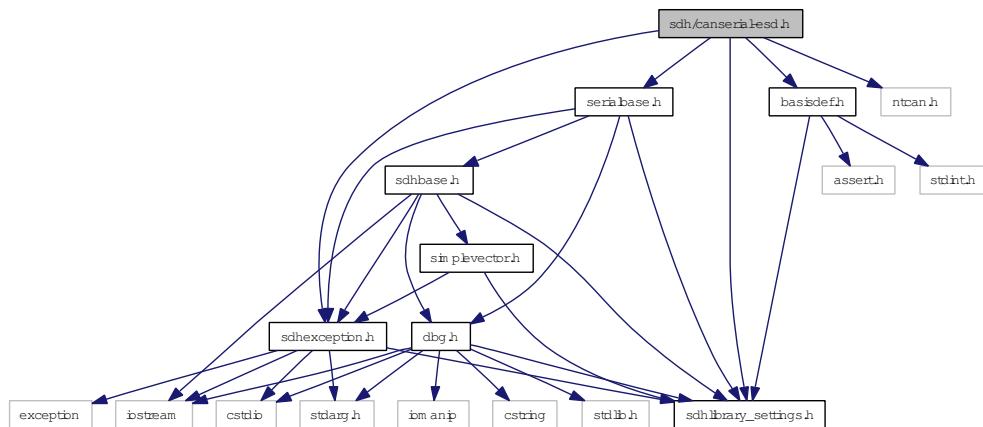
2008-05-02

### 11.42.3 Copyright

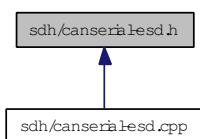
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhexception.h"
#include "serialbase.h"
#include "basisdef.h"
#include "ntcan.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for canserial-esd.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cCANSerial\\_ESDEception](#)  
*Derived exception class for low-level CAN ESD related exceptions.*
- class [SDH::cCANSerial\\_ESD](#)  
*Low-level communication class to access a CAN port.*

## Defines

- #define [NOMINMAX](#)
- #define [CAN\\_ESD\\_TXQUEUESIZE](#) 32  
*transmit queue size for CAN frames*
- #define [CAN\\_ESD\\_RXQUEUESIZE](#) 512  
*receive queue size for CAN frames*

### 11.42.4 Define Documentation

#### 11.42.4.1 #define CAN\_ESD\_RXQUEUESIZE 512

receive queue size for CAN frames

#### 11.42.4.2 #define CAN\_ESD\_TXQUEUESIZE 32

transmit queue size for CAN frames

#### 11.42.4.3 #define NOMINMAX

## 11.43 cpp.desire.final/sdh/canserial-peak.cpp File Reference

### 11.43.1 Detailed Description

Implementation of class [SDH::cCANSerial\\_PEAK](#), a class to access a PEAK CAN interface on cygwin/linux and Visual Studio.

### 11.43.2 General file information

#### Author:

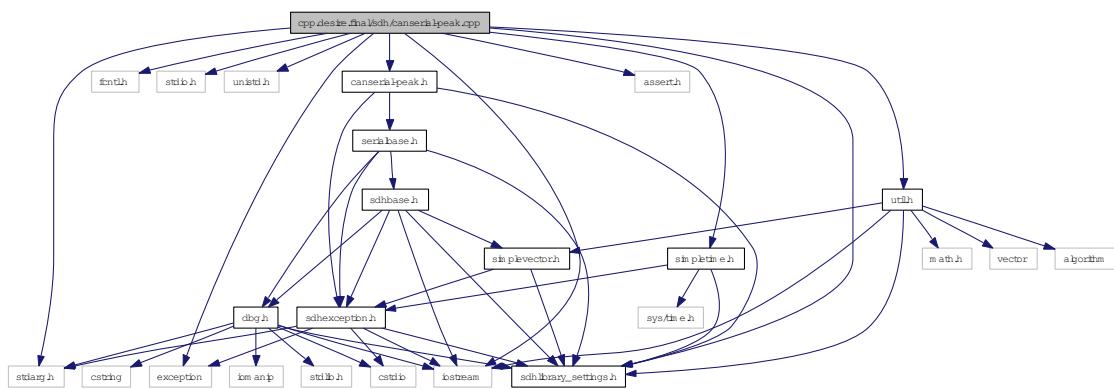
Steffen Ruehl, Dirk Osswald

#### Date:

2009-07-29

```
#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:



### Defines

- #define [USE\\_HANDLES\(H\\_\)](#)
- #define [USE\\_HANDLE\(H\\_\)](#)

## Functions

- char const \* **PEAK\_strerror** (DWORD rc)

### 11.43.3 Define Documentation

11.43.3.1 #define USE\_HANDLE(H\_)

11.43.3.2 #define USE\_HANDLES(H\_)

### 11.43.4 Function Documentation

11.43.4.1 char const\* **PEAK\_strerror** (DWORD *rc*)

## 11.44 sdh/canserial-peak.cpp File Reference

### 11.44.1 Detailed Description

Implementation of class [SDH::cCANSerial\\_PEAK](#), a class to access a PEAK CAN interface on cygwin/linux and Visual Studio.

### 11.44.2 General file information

#### Author:

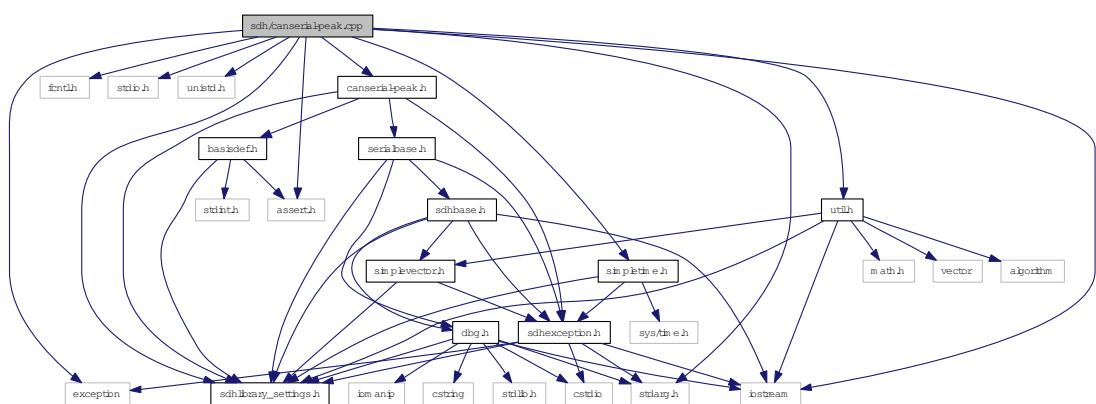
Steffen Ruehl, Dirk Osswald

#### Date:

2009-07-29

```
#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:



### Defines

- #define [USE\\_HANDLES\(H\\_\)](#)
- #define [USE\\_HANDLE\(H\\_\)](#)

## Functions

- char const \* **PEAK\_strerror** (DWORD rc)

### 11.44.3 Define Documentation

11.44.3.1 #define USE\_HANDLE(H\_)

11.44.3.2 #define USE\_HANDLES(H\_)

### 11.44.4 Function Documentation

11.44.4.1 char const\* **PEAK\_strerror** (DWORD *rc*)

## 11.45 cpp.desire.final/sdh/canserial-peak.h File Reference

### 11.45.1 Detailed Description

Interface of class [SDH::cCANSerial\\_Peak](#), class to access CAN bus via PEAK card on cygwin/linux.

### 11.45.2 General file information

#### Author:

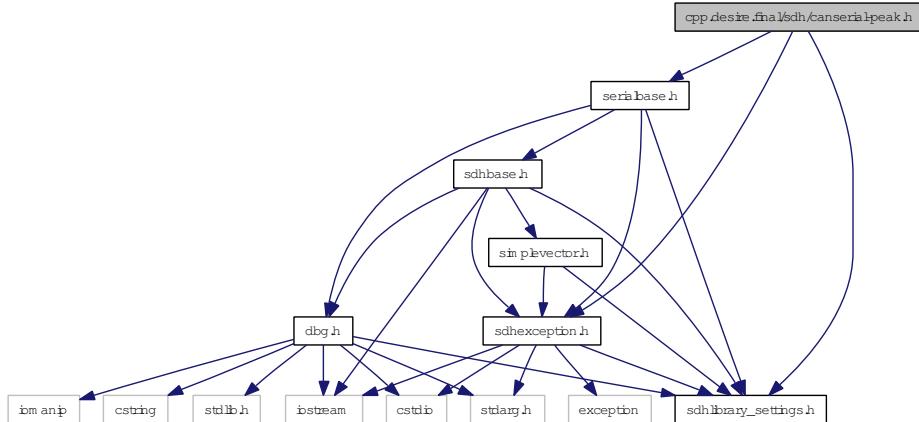
Steffen Ruehl, Dirk Osswald

#### Date:

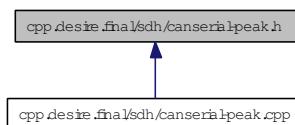
2009-07-29

```
#include "sdhexception.h"
#include "serialbase.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for canserial-peak.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cCANSerial\\_PEAKEception](#)  
*Derived exception class for low-level CAN PEAK related exceptions.*
- class [SDH::cCANSerial\\_PEEK](#)  
*Low-level communication class to access a CAN port.*

## Defines

- #define [M\\_CMSG\\_MSG\(\)](#) m\_cmsg

### 11.45.3 Define Documentation

#### 11.45.3.1 #define M\_CMSG\_MSG() m\_cmsg

## 11.46 sdh/canserial-peak.h File Reference

### 11.46.1 Detailed Description

Interface of class [SDH::cCANSerial\\_PEAK](#), class to access CAN bus via PEAK card on cygwin/linux.

### 11.46.2 General file information

#### Author:

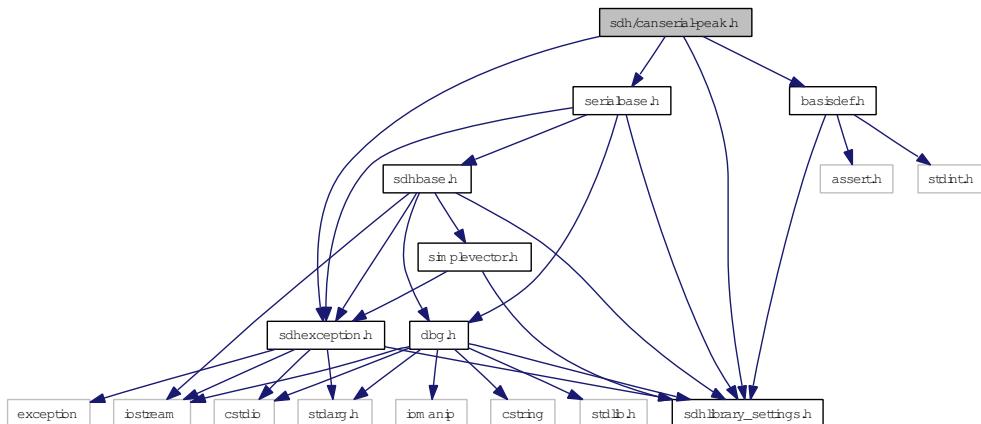
Steffen Ruehl, Dirk Osswald

#### Date:

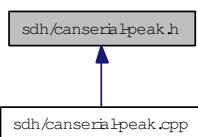
2009-07-29

```
#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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cCANSerial\\_PEAKEception](#)  
*Derived exception class for low-level CAN PEAK related exceptions.*
- class [SDH::cCANSerial\\_PEEK](#)  
*Low-level communication class to access a CAN port.*

## Defines

- #define [M\\_CMSG\\_MSG\(\)](#) m\_cmsg

### 11.46.3 Define Documentation

#### 11.46.3.1 #define M\_CMSG\_MSG() m\_cmsg

## 11.47 cpp.desire.final/sdh/crc.cpp File Reference

### 11.47.1 Detailed Description

Implementation of class `SDH::cCRC_DSACON32m` (actually only the static members all other is derived).

### 11.47.2 General file information

#### Author:

Dirk Osswald

#### Date:

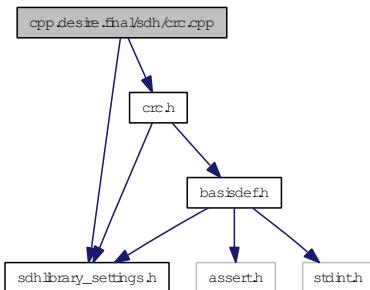
2007-02-19

### 11.47.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include "crc.h"
```

Include dependency graph for crc.cpp:



## 11.48 sdh/crc.cpp File Reference

### 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:

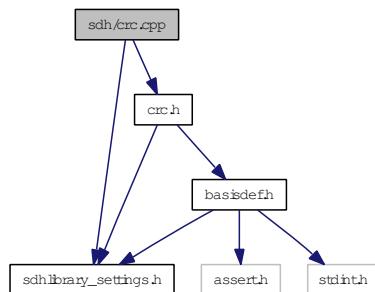
2007-02-19

### 11.48.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"  
#include "crc.h"
```

Include dependency graph for crc.cpp:



## 11.49 cpp.desire.final/sdh/crc.h File Reference

### 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:

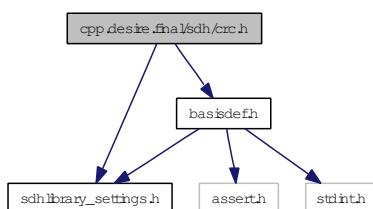
2008-06-09

### 11.49.3 Copyright

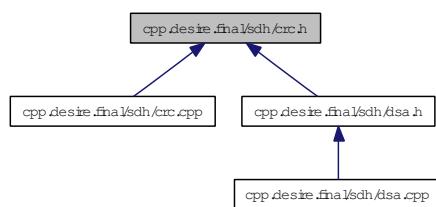
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include "basisdef.h"
```

Include dependency graph for crc.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cCRC](#)

- class [SDH::cCRC\\_DSACON32m](#)

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

## Typedefs

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

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

## 11.50 sdh/crc.h File Reference

### **11.50.1 Detailed Description**

This file contains interface to cCRC, a class to handle CRC calculation.

### **11.50.2 General file information**

**Author:**

Dirk Osswald

Date:

2008-06-09

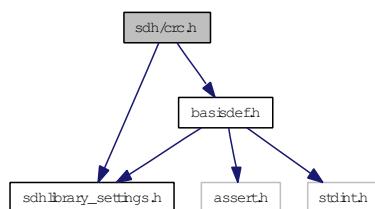
### 11.50.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

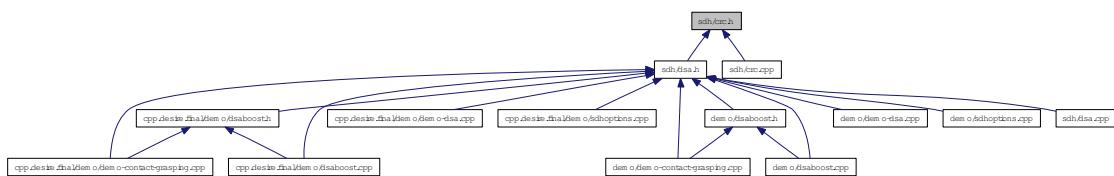
```
#include "sdhlibrary_settings.h"
```

```
#include "basisdef.h"
```

Include dependency graph for crc.h:



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



## Namespaces

- namespace **SDH**

## Classes

- class SDH::cCRC
  - class SDH::cCRC DSACON32m

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

## 11.51 cpp.desire.final/sdh/dbg.h File Reference

### 11.51.1 Detailed Description

This file contains interface and implementation of class **SDH::cDBG**, a class for colorfull debug messages.

### 11.51.2 General file information

#### Author:

Dirk Osswald

#### Date:

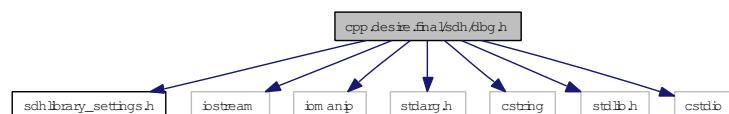
2007-02-22

### 11.51.3 Copyright

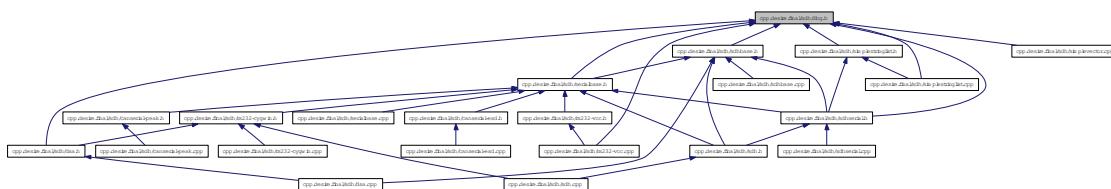
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <iostream>
#include <iomanip>
#include <stdarg.h>
#include <cstring>
#include <stdlib.h>
#include <cstdio>
```

Include dependency graph for dbg.h:



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



## Namespaces

- namespace **SDH**

## Classes

- class [SDH::cDBG](#)  
*A class to print colored debug messages.*

## Defines

- #define **VAR(\_d, \_var)** (\_d) << #\_var << "=" << \_var << "\n"
- #define **V(\_var)** #\_var << "=" << \_var << "

### 11.51.4 Define Documentation

**11.51.4.1 #define V(\_var) #\_var << "=" << \_var << " "**

**11.51.4.2 #define VAR(\_d, \_var) (\_d) << #\_var << "=" << \_var << "\n"**

## 11.52 sdh/dbg.h File Reference

### 11.52.1 Detailed Description

This file contains interface and implementation of class [SDH::cDBG](#), a class for colorfull debug messages.

### 11.52.2 General file information

#### Author:

Dirk Osswald

#### Date:

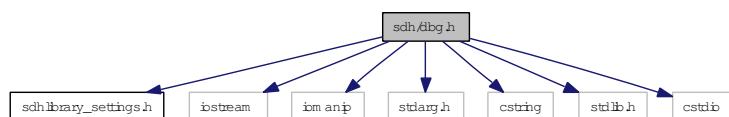
2007-02-22

### 11.52.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <iostream>
#include <iomanip>
#include <stdarg.h>
#include <cstring>
#include <stdlib.h>
#include <cstdio>
```

Include dependency graph for dbg.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cDBG](#)

*A class to print colored debug messages.*

## Defines

- #define **VAR**(\_d, \_var) (\_d) << #\_var << "=" << \_var << "\n"
- #define **V**(\_var) #\_var << "=" << \_var << " "

### 11.52.4 Define Documentation

**11.52.4.1 #define V(\_var) #\_var << "=" << \_var << " "**

**11.52.4.2 #define VAR(\_d, \_var) (\_d) << #\_var << "=" << \_var << "\n"**

## 11.53 cpp.desire.final/sdh/dsa.cpp File Reference

### 11.53.1 Detailed Description

This file contains definition of [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

### 11.53.2 General file information

**Author:**

Dirk Osswald

**Date:**

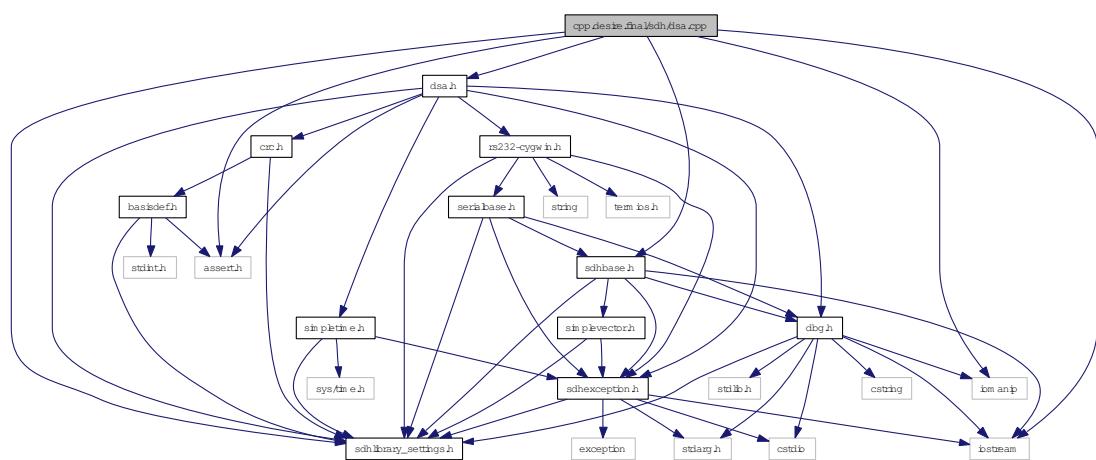
2008-06-09

### 11.53.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <iostream>
#include <iomanip>
#include "dsa.h"
#include "sdhbbase.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"
- #define SDH\_NAMESPACE\_PREFIX SDH::

## Functions

- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sControllerInfo const &controller\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sSensorInfo const &sensor\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sMatrixInfo const &matrix\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sResponse const &response)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA const &dsa)

### 11.53.4 Define Documentation

11.53.4.1 #define PRINT\_MEMBER(\_s, \_var, \_member) (\_s) << " " << #\_member << "=" << \_var.\_member << "\n"

11.53.4.2 #define PRINT\_MEMBER\_HEX(\_s, \_var, \_member) (\_s) << " " << #\_member << "=0x" << std::hex << int(\_var.\_member) << std::dec << "\n"

11.53.4.3 #define SDH\_NAMESPACE\_PREFIX SDH::

## 11.54 sdh/dsa.cpp File Reference

### 11.54.1 Detailed Description

This file contains definition of [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

### 11.54.2 General file information

**Author:**

Dirk Osswald

**Date:**

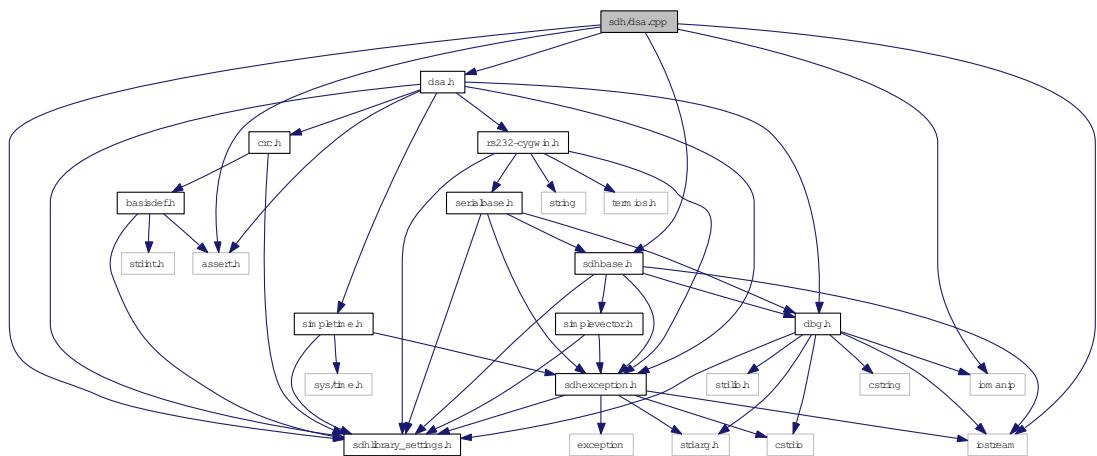
2008-06-09

### 11.54.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <iostream>
#include <iomanip>
#include "dsa.h"
#include "sdhbbase.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"
- #define SDH\_NAMESPACE\_PREFIX SDH::

## Functions

- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sControllerInfo const &controller\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sSensorInfo const &sensor\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sMatrixInfo const &matrix\_info)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sResponse const &response)
- std::ostream & SDH::operator<< (std::ostream &stream, cDSA const &dsa)

### 11.54.4 Define Documentation

11.54.4.1 #define PRINT\_MEMBER(\_s, \_var, \_member) (\_s) << " " << #\_member << "=" << \_var.\_member << "\n"

11.54.4.2 #define PRINT\_MEMBER\_HEX(\_s, \_var, \_member) (\_s) << " " << #\_member << "=0x" << std::hex << int(\_var.\_member) << std::dec << "\n"

11.54.4.3 #define SDH\_NAMESPACE\_PREFIX SDH::

## 11.55 cpp.desire.final/sdh/dsa.h File Reference

### 11.55.1 Detailed Description

This file contains interface to [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

### 11.55.2 General file information

**Author:**

Dirk Osswald

**Date:**

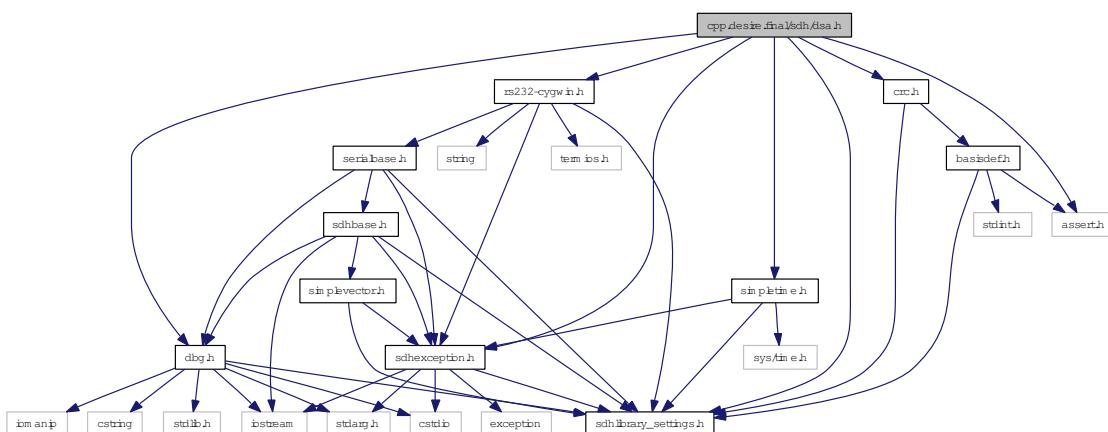
2008-06-09

### 11.55.3 Copyright

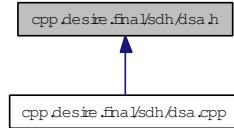
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## 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::sTactileSensorFrame](#)
- 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*

## Defines

- #define [DSA\\_MAX\\_PREAMBLE\\_SEARCH](#) (2\*3\*(6\*(14+13)) + 16)

## Functions

- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sControllerInfo](#) const &controller\_info)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sSensorInfo](#) const &sensor\_info)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sMatrixInfo](#) const &matrix\_info)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA::sResponse](#) const &response)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cDSA](#) const &dsa)

## 11.55.4 Define Documentation

11.55.4.1 `#define DSA_MAX_PREAMBLE_SEARCH (2*3*(6*(14+13)) + 16)`

## 11.56 sdh/dsa.h File Reference

### 11.56.1 Detailed Description

This file contains interface to [SDH::cDSA](#), a class to communicate with the tactile sensors of the SDH.

### 11.56.2 General file information

#### Author:

Dirk Osswald

#### Date:

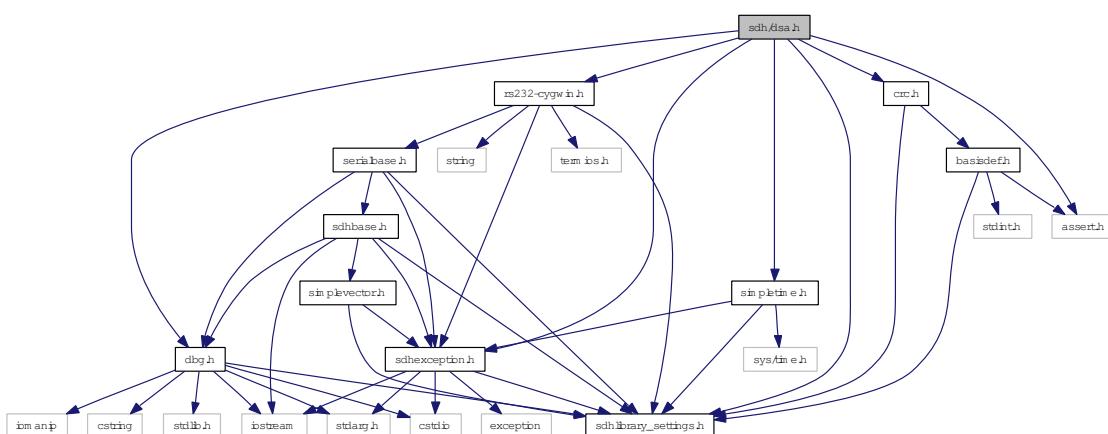
2008-06-09

### 11.56.3 Copyright

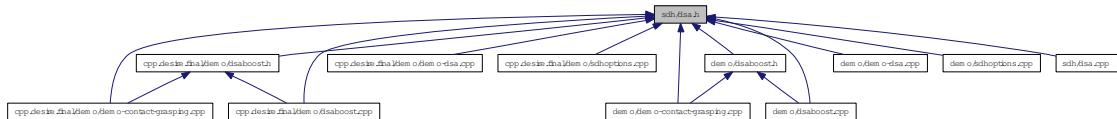
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace **SDH**

## 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::sTactileSensorFrame

- 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*

## Defines

- #define DSA\_MAX\_PREAMBLE\_SEARCH (2\*3\*(6\*(14+13)) + 16)

## Functions

- std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sControllerInfo const &controller\_info)
  - std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sSensorInfo const &sensor\_info)
  - std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sMatrixInfo const &matrix\_info)
  - std::ostream & SDH::operator<< (std::ostream &stream, cDSA::sResponse const &response)
  - std::ostream & SDH::operator<< (std::ostream &stream, cDSA const &ds)

## 11.56.4 Define Documentation

11.56.4.1 `#define DSA_MAX_PREAMBLE_SEARCH (2*3*(6*(14+13)) + 16)`

## 11.57 cpp.desire.final/sdh/release.h File Reference

### 11.57.1 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.

### 11.57.2 General file information

#### Author:

Dirk Osswald

#### Date:

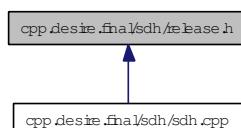
2006-11-30

For a general description of the project see [general project information](#).

### 11.57.3 Copyright

Copyright (c) 2006 SCHUNK GmbH & Co. KG

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



### Defines

- #define `PROJECT_NAME` "SDHLibrary-CPP"

*Name of the software project.*

- #define `PROJECT_RELEASE` "0.0.1.11"

*Release name of the whole software project (a.k.a. as the "version" of the project).*

- #define `PROJECT_DATE` "2009-11-02"

*Date of the release of the software project.*

- #define `PROJECT_COPYRIGHT` "(c) SCHUNK GmbH & Co. KG, 2007"

## 11.57.4 Define Documentation

**11.57.4.1 #define PROJECT\_COPYRIGHT "(c) SCHUNK GmbH & Co. KG, 2007"**

**11.57.4.2 #define PROJECT\_DATE "2009-11-02"**

Date of the release of the software project.

The date of the release of the project.

**11.57.4.3 #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.

**11.57.4.4 #define PROJECT\_RELEASE "0.0.1.11"**

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:

- **0.0.1.11:**

- 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
- 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

- **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` reopend 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 Bergström.
  - 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: <a href="[https://192.168.101.101/mechatronik/show\\_bug.cgi?id=478](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 cancat 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)
    - \* sdhlibraryDefines 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 interface member to comm\_interface
- **0.0.0.7-a:** 2008-05-17
  - bug fix: minor changes to make compilation work on Linux. (ESDs ntcan.h for Windows is different from that for Linux)
  - still contains a bug that can be fixed without recompilation:
    - \* The Linux version of ntcan 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 cSerialBaseException, cCANSerialESDEception)
  - 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 to 20,0,0,0,0,0:
    - . axis 1 is stuck at 1.4...
    - . bug could not be resolved (does not happen for larger movements)
- **0.0.4:** 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)
  - **0.0.3-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
  - **0.0.3:** 2007-03-09

- Release modified at visit Uni-Whales
  - \* 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 relyably

## 11.58 `sdh/release.h` File Reference

### 11.58.1 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.

### 11.58.2 General file information

#### Author:

Dirk Osswald

#### Date:

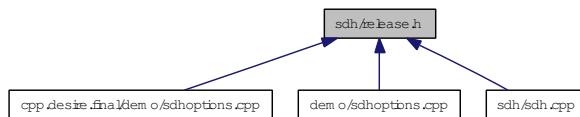
2006-11-30

For a general description of the project see [general project information](#).

### 11.58.3 Copyright

Copyright (c) 2006 SCHUNK GmbH & Co. KG

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



### Defines

- `#define PROJECT_NAME "SDHLibrary-CPP"`  
*Name of the software project.*
- `#define PROJECT_RELEASE "0.0.1.13"`  
*Release name of the whole software project (a.k.a. as the "version" of the project).*
- `#define PROJECT_DATE "2010-02-02"`  
*Date of the release of the software project.*
- `#define PROJECT_COPYRIGHT "(c) SCHUNK GmbH & Co. KG, 2007"`

### 11.58.4 Define Documentation

#### 11.58.4.1 `#define PROJECT_COPYRIGHT "(c) SCHUNK GmbH & Co. KG, 2007"`

#### 11.58.4.2 `#define PROJECT_DATE "2010-02-02"`

Date of the release of the software project.

The date of the release of the project.

#### 11.58.4.3 #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.

#### 11.58.4.4 #define PROJECT\_RELEASE "0.0.1.13"

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:

- **0.0.1.13:** (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: SDHLibrary 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 reopend 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 Bergström.
- 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: <a href="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 cancat 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)
    - \* sdhlibraryDefines 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 interface member to comm\_interface
  - **0.0.0.7-a:** 2008-05-17
    - bug fix: minor changes to make compilation work on Linux. (ESDs ntcan.h for Windows is different from that for Linux)
    - still contains a bug that can be fixed without recompilation:
      - \* The Linux version of ntcan 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 cSerialBaseException, cCANSerialESDEception)
    - 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 to 20,0,0,0,0,0:
    - . 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-Whales
    - \* 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

## 11.59 cpp.desire.final/sdh/rs232-cygwin.cpp File Reference

### 11.59.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

### 11.59.2 General file information

#### Author:

Dirk Osswald

#### Date:

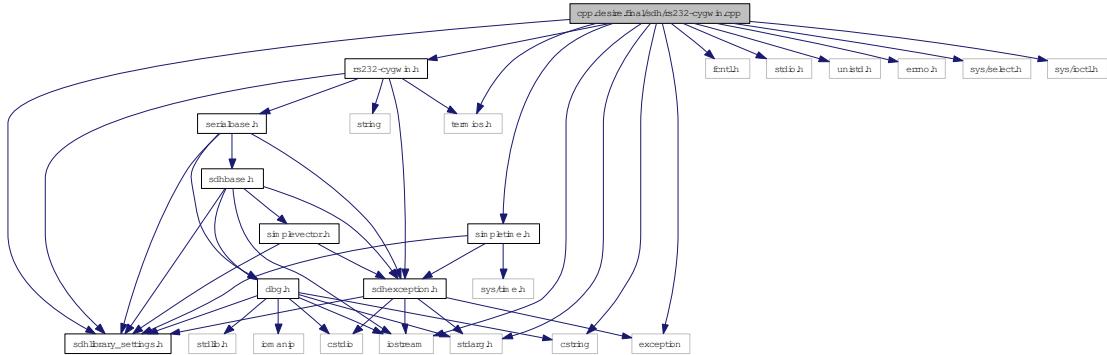
2007-02-20

### 11.59.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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 <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 SDH_RS232_CYGWIN_DEBUG_ASCII 0`
- `#define DBG(...);`

## Functions

- `char * StrDupNew (char *const s)`  
*helper function, duplicate string s into a char array allocated with new[]*

### 11.59.4 Define Documentation

#### 11.59.4.1 `#define DBG( ...);`

#### 11.59.4.2 `#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.

#### 11.59.4.3 `#define SDH_RS232_CYGWIN_DEBUG_ASCII 0`

Flag, if true then ascii codes of data bytes sent/received are printed too if `SDH_RS232_CYGWIN_DEBUG` is true and the debug level is high enough.

## 11.59.5 Function Documentation

#### 11.59.5.1 `char* StrDupNew (char *const s)`

helper function, duplicate string s into a char array allocated with `new[]`

## 11.60 **sdh/rs232-cygwin.cpp** File Reference

### 11.60.1 Detailed Description

Implementation of class **SDH::cRS232**, a class to access serial RS232 port on cygwin/linux.

### 11.60.2 General file information

**Author:**

Dirk Osswald

**Date:**

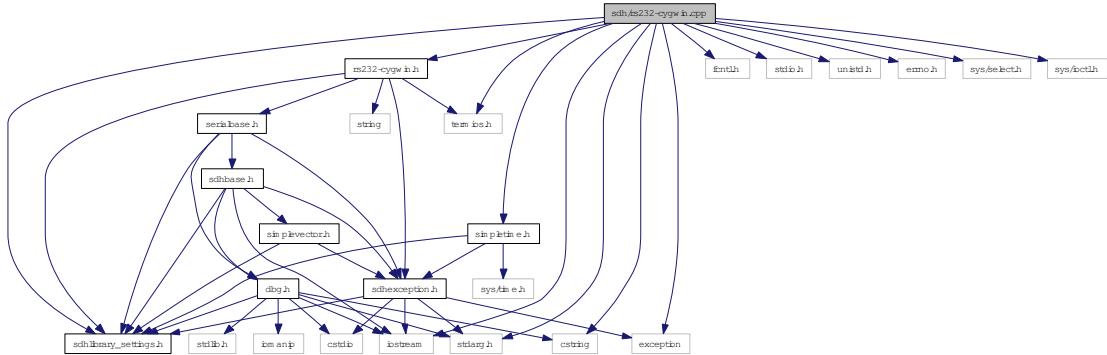
2007-02-20

### 11.60.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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 <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 **SDH\_RS232\_CYGWIN\_DEBUG\_ASCII** 0
- #define **DBG(...);**

## Functions

- char \* **StrDupNew** (char \*const s)  
*helper function, duplicate string s into a char array allocated with new[]*

### 11.60.4 Define Documentation

#### 11.60.4.1 #define **DBG(...);**

#### 11.60.4.2 #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.debug.SetFlag(1)`.

This 2 level scheme is used since this is the lowlevel communication, so debug outputs might really steal some performance.

#### 11.60.4.3 #define **SDH\_RS232\_CYGWIN\_DEBUG\_ASCII** 0

Flag, if true then ascii codes of data bytes sent/received are printed too if `SDH_RS232_CYGWIN_DEBUG` is true and the debug level is high enough.

### 11.60.5 Function Documentation

#### 11.60.5.1 char\* **StrDupNew** (char \*const s)

helper function, duplicate string s into a char array allocated with new[]

## 11.61 cpp.desire.final/sdh/rs232-cygwin.h File Reference

### 11.61.1 Detailed Description

Interface of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

### 11.61.2 General file information

#### Author:

Dirk Osswald

#### Date:

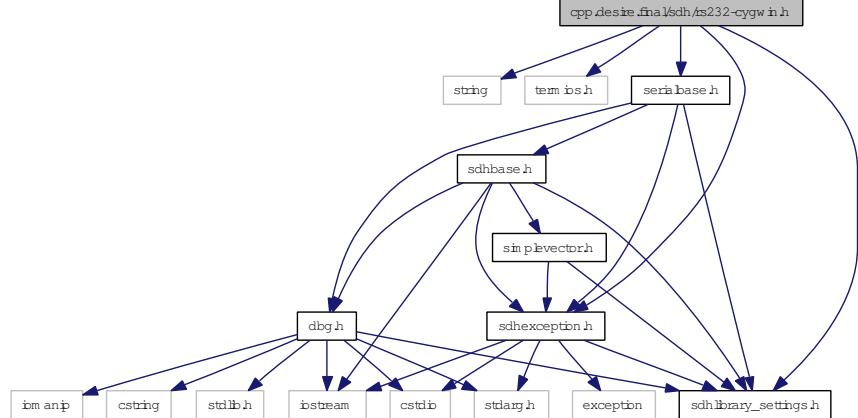
2007-02-20

### 11.61.3 Copyright

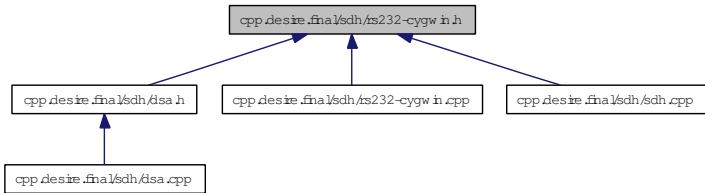
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## 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.*

## 11.62 sdh/rs232-cygwin.h File Reference

### 11.62.1 Detailed Description

Interface of class [SDH::cRS232](#), a class to access serial RS232 port on cygwin/linux.

### 11.62.2 General file information

**Author:**

Dirk Osswald

**Date:**

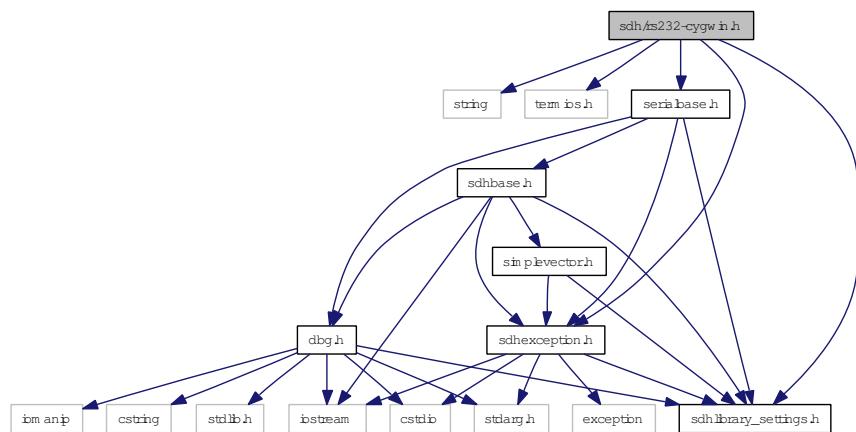
2007-02-20

### 11.62.3 Copyright

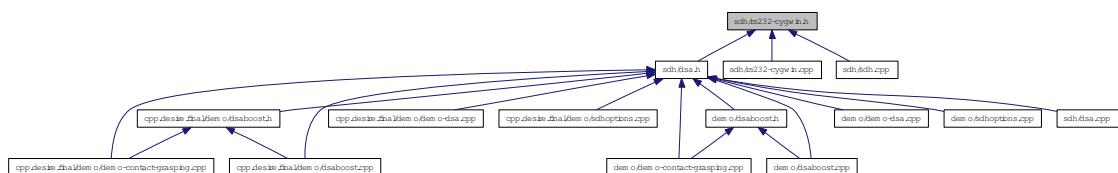
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## 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.*

## 11.63 cpp.desire.final/sdh/rs232-vcc.cpp File Reference

### 11.63.1 Detailed Description

Implementation of class `SDH::cRS232`, a class to access serial RS232 port with VCC compiler on Windows.

### 11.63.2 General file information

#### Author:

Martin

#### Date:

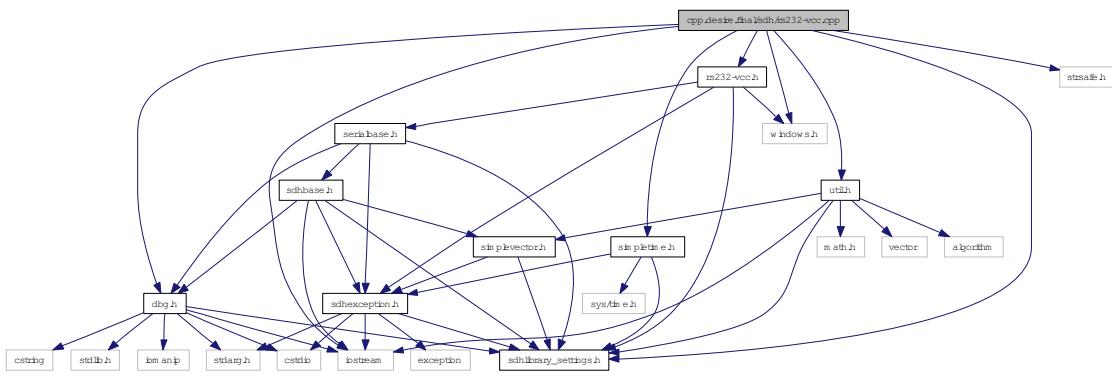
2008-05-23

### 11.63.3 Copyright

Code kindly provided by Martin from the RoboCluster project Denmark.

```
#include "iostream"
#include <windows.h>
#include <strsafe.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 SDH\_RS232\_VCC\_DEBUG\_ASCII 0
- #define DBG(...) \_\_VA\_ARGS\_\_

## Functions

- char const \* **GetLastErrorMessage** (void)

*return the last windows error message as string. The string returned will be overwritten by the next call to the function*

### 11.63.4 Define Documentation

#### 11.63.4.1 #define \_CRT\_SECURE\_NO\_WARNINGS 1

#### 11.63.4.2 #define DBG( ...) \_\_VA\_ARGS\_\_

#### 11.63.4.3 #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.

#### 11.63.4.4 #define SDH\_RS232\_VCC\_DEBUG\_ASCII 0

Flag, if true then ascii codes of data bytes sent/received are printed too if SDH\_RS232\_CYGWIN\_DEBUG is true and the debug level is high enough.

### 11.63.5 Function Documentation

#### 11.63.5.1 char const\* GetLastErrorMessage (void)

return the last windows error message as string. The string returned will be overwritten by the next call to the function

## 11.64 `sdh/rs232-vcc.cpp` File Reference

### 11.64.1 Detailed Description

Implementation of class `SDH::cRS232`, a class to access serial RS232 port with VCC compiler on Windows.

### 11.64.2 General file information

#### Author:

Martin

#### Date:

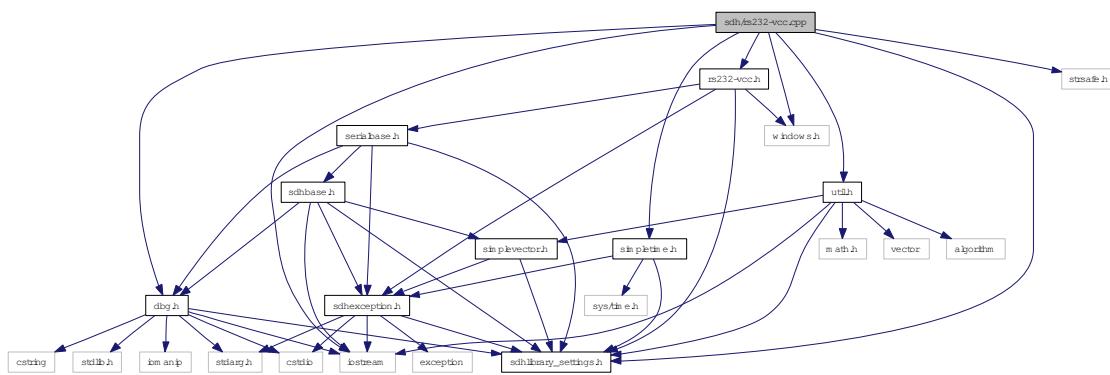
2008-05-23

### 11.64.3 Copyright

Code kindly provided by Martin from the RoboCluster project Denmark.

```
#include "iostream"
#include <windows.h>
#include <strsafe.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 SDH\_RS232\_VCC\_DEBUG\_ASCII 0
- #define DBG(...) \_\_VA\_ARGS\_\_

## Functions

- char const \* **GetLastErrorMessage** (void)

*return the last windows error message as string. The string returned will be overwritten by the next call to the function*

### 11.64.4 Define Documentation

#### 11.64.4.1 #define \_CRT\_SECURE\_NO\_WARNINGS 1

#### 11.64.4.2 #define DBG( ...) \_\_VA\_ARGS\_\_

#### 11.64.4.3 #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.

#### 11.64.4.4 #define SDH\_RS232\_VCC\_DEBUG\_ASCII 0

Flag, if true then ascii codes of data bytes sent/received are printed too if SDH\_RS232\_CYGWIN\_DEBUG is true and the debug level is high enough.

### 11.64.5 Function Documentation

#### 11.64.5.1 char const\* GetLastErrorMessage (void)

return the last windows error message as string. The string returned will be overwritten by the next call to the function

## 11.65 cpp.desire.final/sdh/rs232-vcc.h File Reference

### 11.65.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

### 11.65.2 General file information

**Author:**

Martin

**Date:**

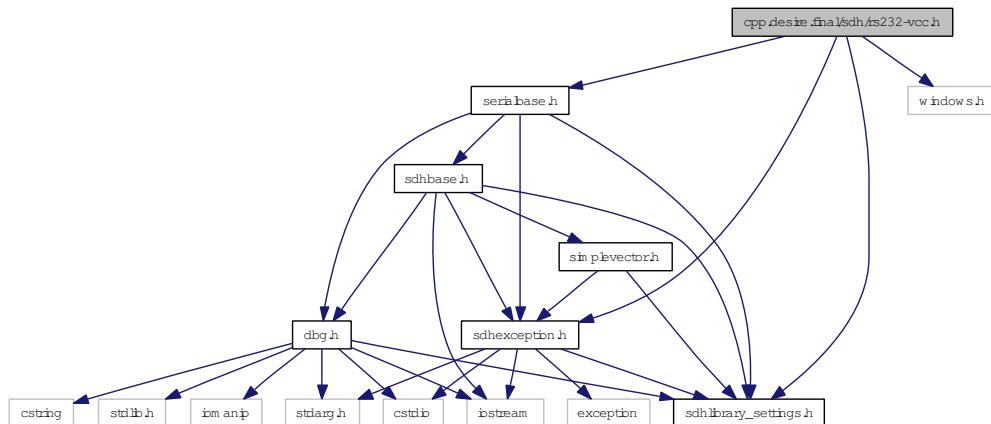
2008-05-23

### 11.65.3 Copyright

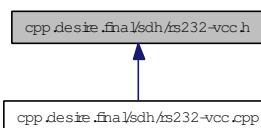
Code kindly provided by Martin from the RoboCluster project Denmark.

```
#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:



## Namespaces

- namespace [SDH](#)

## 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.*

## Defines

- #define [SDH\\_RS232\\_VCC\\_ASYNC](#) 0

### 11.65.4 Define Documentation

#### 11.65.4.1 #define SDH\_RS232\_VCC\_ASYNC 0

## 11.66 sdh/rs232-vcc.h File Reference

### 11.66.1 Detailed Description

Implementation of class [SDH::cRS232](#), a class to access serial RS232 port with VCC compiler on Windows.

### 11.66.2 General file information

#### Author:

Martin

#### Date:

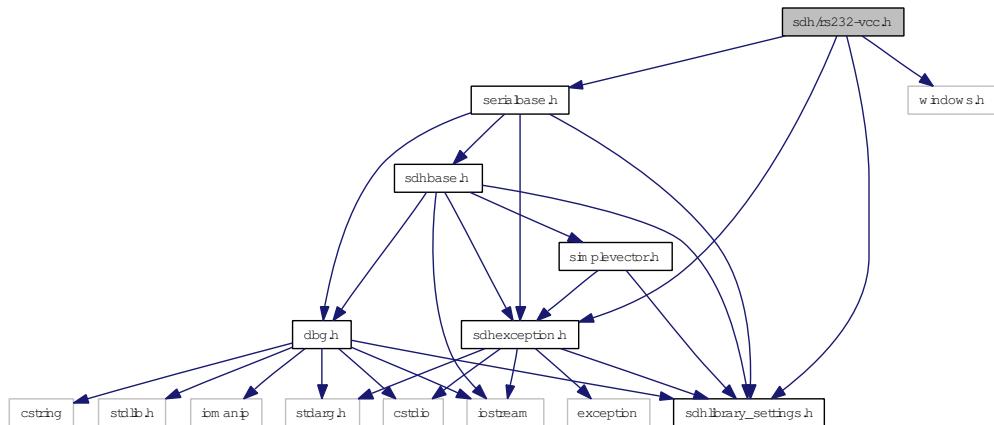
2008-05-23

### 11.66.3 Copyright

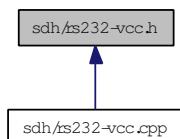
Code kindly provided by Martin from the RoboCluster project Denmark.

```
#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:



## Namespaces

- namespace [SDH](#)

## 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.*

## Defines

- #define [SDH\\_RS232\\_VCC\\_ASYNC](#) 0

### 11.66.4 Define Documentation

#### 11.66.4.1 #define SDH\_RS232\_VCC\_ASYNC 0

## **11.67 cpp.desire.final/sdh/sdh.cpp File Reference**

### **11.67.1 Detailed Description**

This file contains the interface to class **SDH::cSDH**, the end user class to access the SDH from a PC.

### **11.67.2 General file information**

**Author:**

Dirk Osswald

Date:

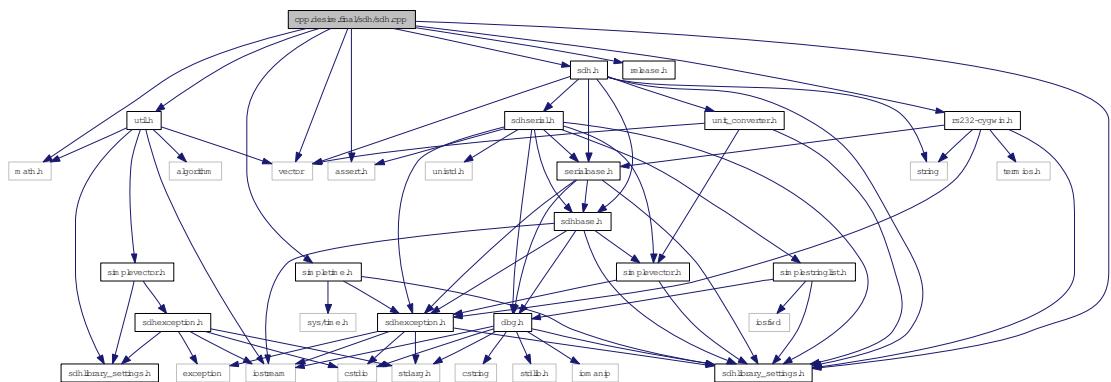
2007-02-20

### 11.67.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"  
#include <assert.h>  
#include <vector>  
#include <math.h>  
#include <util.h>  
#include "sdh.h"  
#include "release.h"  
#include "simplesoftware.h"  
#include "rs232-cygwin.h"
```

Include dependency graph for sdh.cpp:



## 11.68 sdh/sdh.cpp File Reference

### 11.68.1 Detailed Description

This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

### 11.68.2 General file information

#### Author:

Dirk Osswald

#### Date:

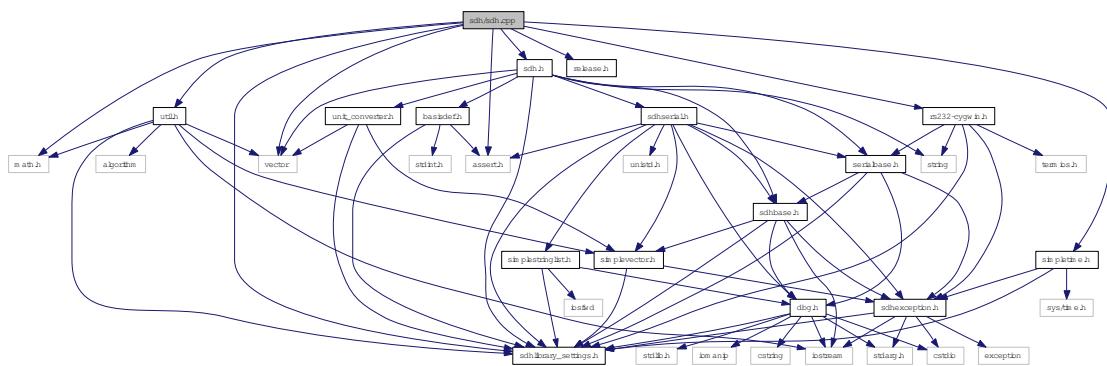
2007-02-20

### 11.68.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include <math.h>
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <vector>
#include <util.h>
#include "sdh.h"
#include "release.h"
#include "simpletime.h"
#include "rs232-cygwin.h"
```

Include dependency graph for sdh.cpp:



### Defines

- #define [\\_USE\\_MATH\\_DEFINES](#)

## 11.68.4 Define Documentation

### 11.68.4.1 #define \_USE\_MATH\_DEFINES

## 11.69 cpp.desire.final/sdh/sdh.h File Reference

### 11.69.1 Detailed Description

This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

### 11.69.2 General file information

#### Author:

Dirk Osswald

#### Date:

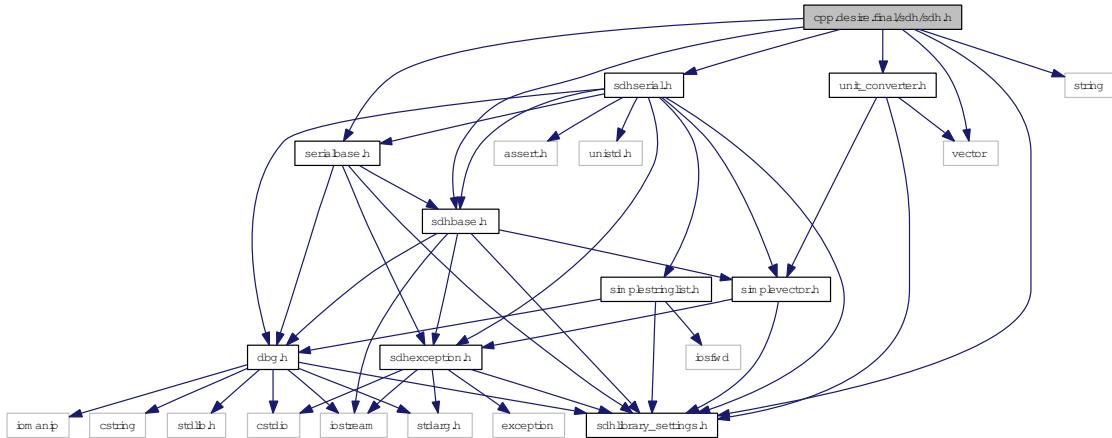
2007-02-20

### 11.69.3 Copyright

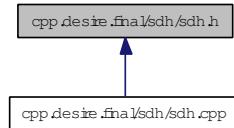
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <vector>
#include <string>
#include "sdhbbase.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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDH](#)

*SDH::cSDH is the end user interface class to control a SDH (SCHUNK Dexterous Hand).*

## Typedefs

- typedef void \* [SDH::NTCAN\\_HANDLE](#)

*dummy definition in case ntcan.h is not available*

## 11.70 sdh/sdh.h File Reference

### 11.70.1 Detailed Description

This file contains the interface to class [SDH::cSDH](#), the end user class to access the SDH from a PC.

### 11.70.2 General file information

#### Author:

Dirk Osswald

#### Date:

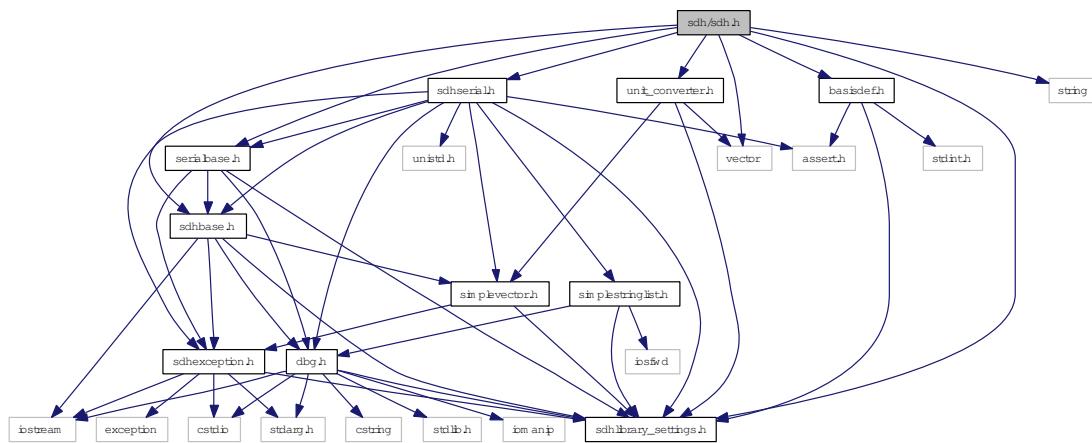
2007-02-20

### 11.70.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include "basisdef.h"
#include <vector>
#include <string>
#include "sdhbbase.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:

## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDH](#)

*SDH::cSDH is the end user interface class to control a SDH (SCHUNK Dexterous Hand).*

## Typedefs

- typedef void \* [SDH::PCAN\\_HANDLE](#)

*dummy definition in case Pcan\_usb.h is not available*

## 11.71 cpp.desire.final/sdh/sdhbase.cpp File Reference

### 11.71.1 Detailed Description

Implementation of class [SDH::cSDHBase](#).

### 11.71.2 General file information

#### Author:

Dirk Osswald

#### Date:

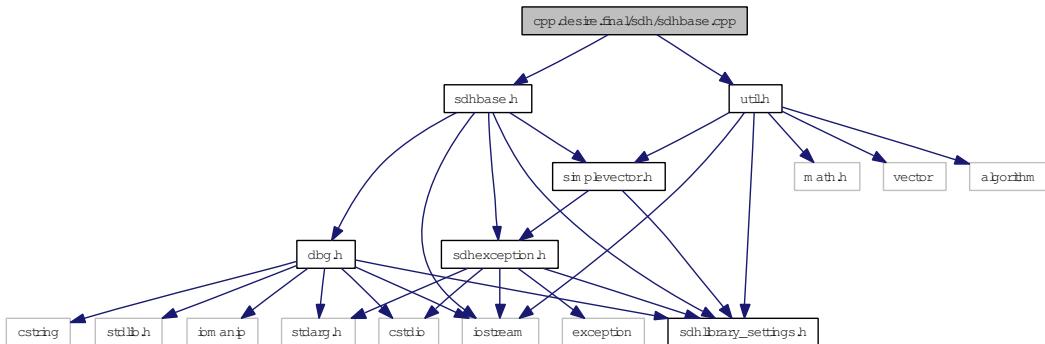
2007-02-19

### 11.71.3 Copyright

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

```
#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`

## 11.72 sdh/sdhbase.cpp File Reference

### 11.72.1 Detailed Description

Implementation of class [SDH::cSDHBase](#).

### 11.72.2 General file information

**Author:**

Dirk Osswald

**Date:**

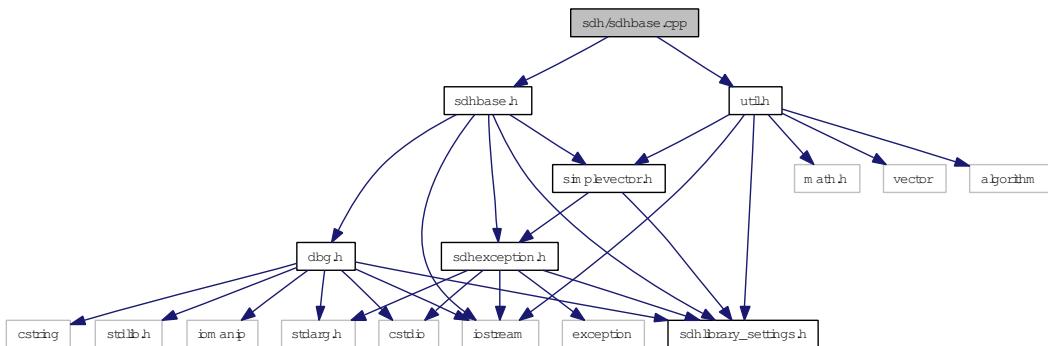
2007-02-19

### 11.72.3 Copyright

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

```
#include "sdhbase.h"
#include "util.h"
```

Include dependency graph for sdhbase.cpp:



## Namespaces

- namespace [SDH](#)

## 11.73 `cpp.desire.final/sdh/sdhbase.h` File Reference

### **11.73.1 Detailed Description**

Interface of class **SDH::cSDHBase**.

### **11.73.2 General file information**

**Author:**

Dirk Osswald

Date:

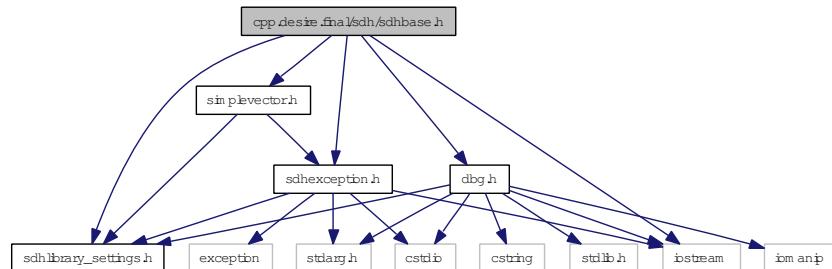
2007-02-19

### 11.73.3 Copyright

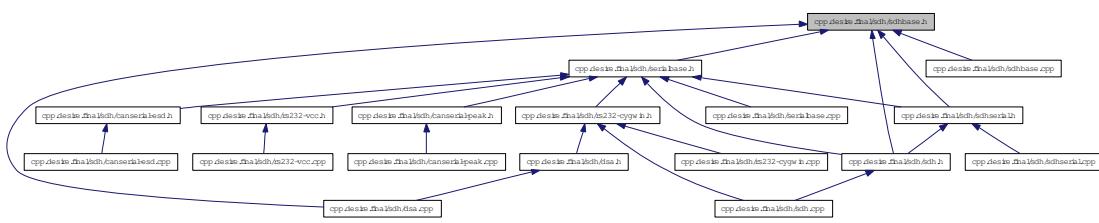
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"  
#include "iostream"  
#include "dbg.h"  
#include "sdhexception.h"  
#include "simplevector.h"
```

Include dependency graph for sdhbase.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDHErrorInvalidParameter](#)  
*Derived exception class for exceptions related to invalid parameters.*
- class [SDH::cSDHBase](#)  
*The base class to control the SCHUNK Dexterous Hand.*

## 11.74 sdh/sdhbase.h File Reference

### 11.74.1 Detailed Description

Interface of class [SDH::cSDHBase](#).

### 11.74.2 General file information

**Author:**

Dirk Osswald

**Date:**

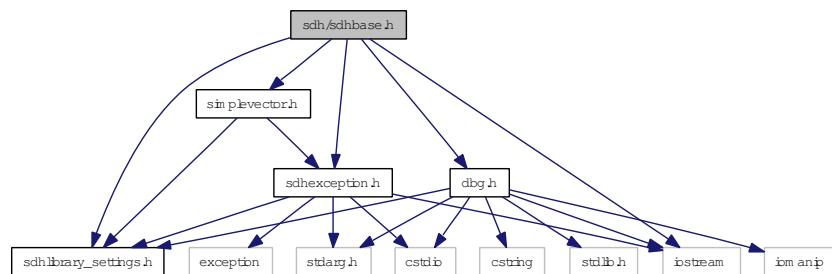
2007-02-19

### 11.74.3 Copyright

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

```
#include "sdhlibrary_settings.h"
#include "iostream"
#include "dbg.h"
#include "sdhexception.h"
#include "simplevector.h"
```

Include dependency graph for sdhbase.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDHErrorInvalidParameter](#)

*Derived exception class for exceptions related to invalid parameters.*

- class [SDH::cSDHBase](#)

*The base class to control the SCHUNK Dexterous Hand.*

## 11.75 cpp.desire.final/sdh/sdhexception.cpp File Reference

### 11.75.1 Detailed Description

Implementation of the exception base class `SDH::cSDHLibraryException` and `SDH::cMsg`.

### 11.75.2 General file information

**Author:**

Dirk Osswald

**Date:**

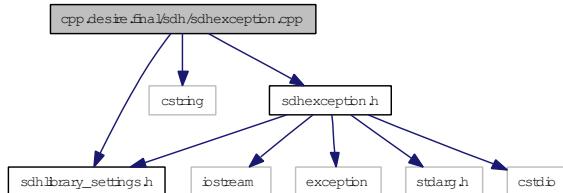
2007-02-22

### 11.75.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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)`

## 11.76 `sdh/sdhexception.cpp` File Reference

### 11.76.1 Detailed Description

Implementation of the exception base class `SDH::cSDHLIBRARYException` and `SDH::cMsg`.

### 11.76.2 General file information

**Author:**

Dirk Osswald

**Date:**

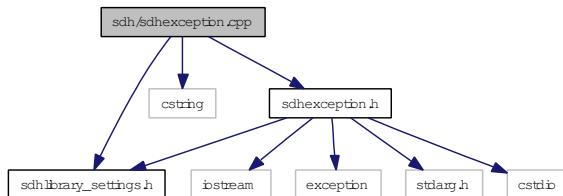
2007-02-22

### 11.76.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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)`

## 11.77 cpp.desire.final/sdh/sdhexception.h File Reference

### 11.77.1 Detailed Description

Interface of the exception base class [SDH::cSDHLibraryException](#) and [SDH::cMsg](#).

### 11.77.2 General file information

#### Author:

Dirk Osswald

#### Date:

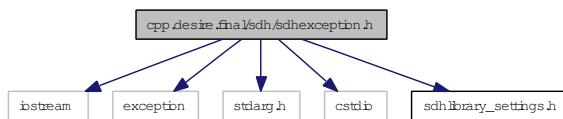
2007-02-22

### 11.77.3 Copyright

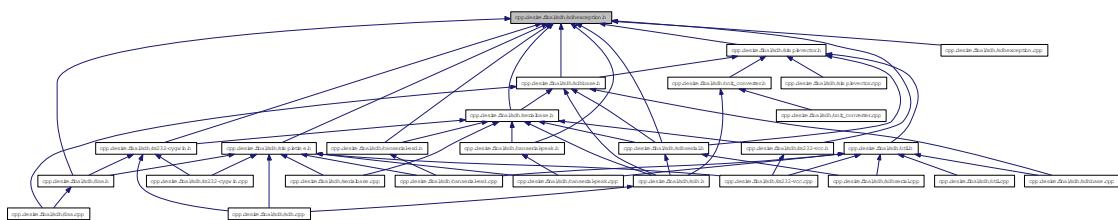
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## 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.*

## Functions

- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cMsg](#) const &msg)
- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cSDHLibraryException](#) const &e)

## 11.78 `sdh/sdhexception.h` File Reference

### 11.78.1 Detailed Description

Interface of the exception base class `SDH::cSDHLlibraryException` and `SDH::cMsg`.

### 11.78.2 General file information

**Author:**

Dirk Osswald

**Date:**

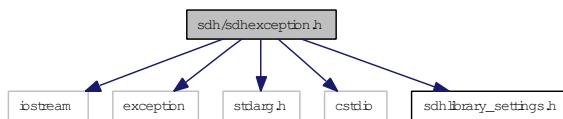
2007-02-22

### 11.78.3 Copyright

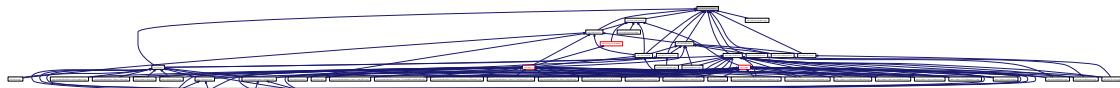
Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace `SDH`

## 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.*

## Functions

- std::ostream & **SDH::operator<<** (std::ostream &stream, **cMsg** const &msg)
- std::ostream & **SDH::operator<<** (std::ostream &stream, **cSDHLibraryException** const &e)

## 11.79 `cpp.desire.final/sdh/sdhlibrary_settings.h` File Reference

### 11.79.1 Detailed Description

This file contains settings to make the SDHLibrary compile on different systems:

- gcc/Cygwin/Windows
- gcc/Linux
- VisualC++/Windows.

### 11.79.2 General file information

#### Author:

Dirk Osswald

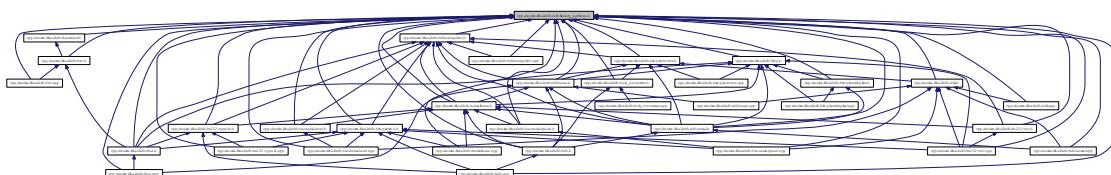
#### Date:

2008-05-20

### 11.79.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

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

## 11.80 `sdh/sdhlibrary_settings.h` File Reference

### 11.80.1 Detailed Description

This file contains settings to make the SDHLibrary compile on different systems:

- gcc/Cygwin/Windows
- gcc/Linux
- VisualC++/Windows.

### 11.80.2 General file information

#### Author:

Dirk Osswald

#### Date:

2008-05-20

### 11.80.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

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

## 11.81 cpp.desire.final/sdh/sdhserial.cpp File Reference

### 11.81.1 Detailed Description

Interface of class [SDH::cSDHSerial](#).

### 11.81.2 General file information

#### Author:

Dirk Osswald

#### Date:

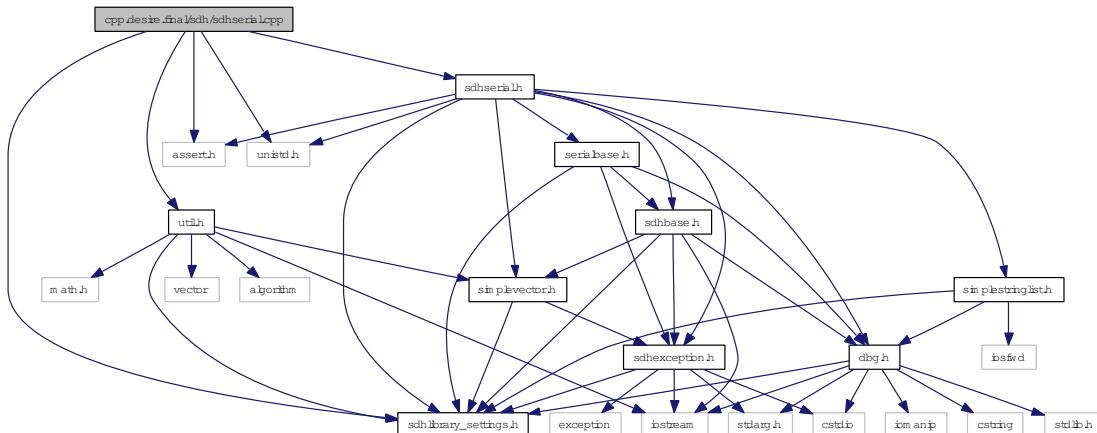
2007-02-19

### 11.81.3 Copyright

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

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "util.h"
#include "sdhserial.h"
```

Include dependency graph for sdhserial.cpp:



## 11.82 sdh/sdhserial.cpp File Reference

### 11.82.1 Detailed Description

Interface of class [SDH::cSDHSerial](#).

### 11.82.2 General file information

**Author:**

Dirk Osswald

**Date:**

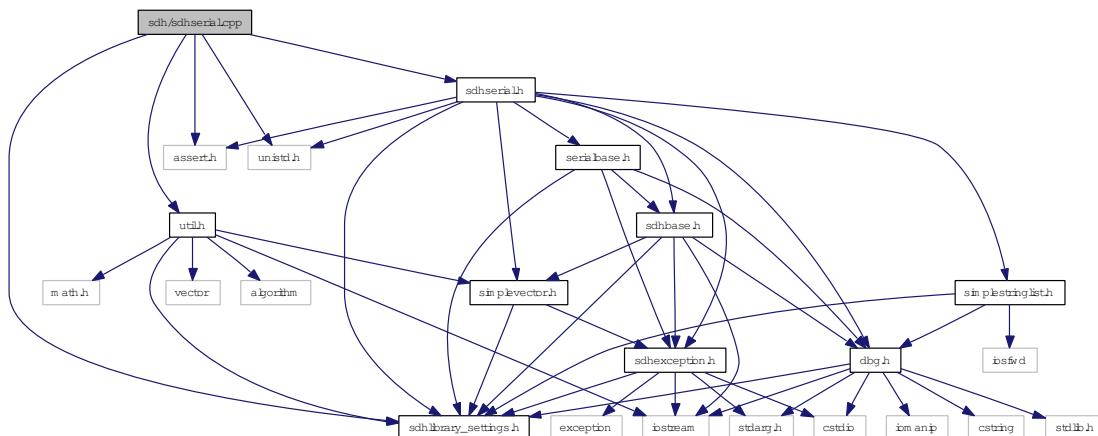
2007-02-19

### 11.82.3 Copyright

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

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "util.h"
#include "sdhserial.h"
```

Include dependency graph for sdhserial.cpp:



## 11.83 cpp.desire.final/sdh/sdhserial.h File Reference

### 11.83.1 Detailed Description

Interface of class [SDH::cSDHSerial](#).

### 11.83.2 General file information

#### Author:

Dirk Osswald

#### Date:

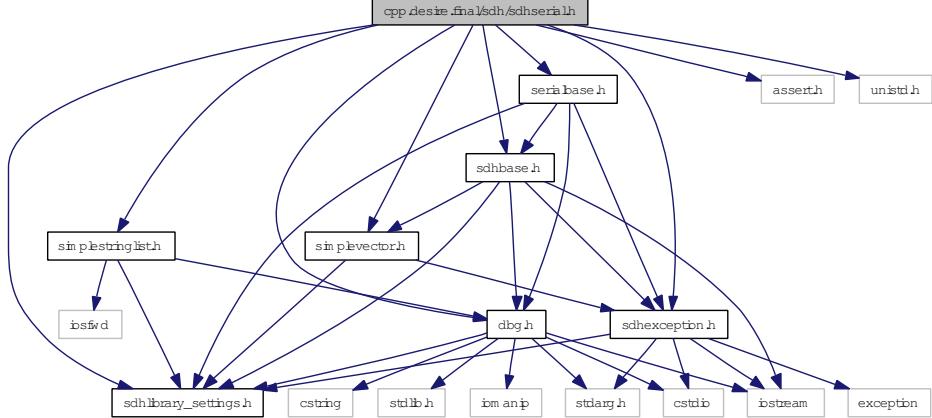
2007-02-19

### 11.83.3 Copyright

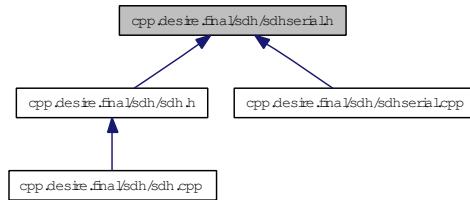
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "dbg.h"
#include "sdhexception.h"
#include "simplevector.h"
#include "simplestringlist.h"
#include "sdhbbase.h"
#include "serialbase.h"
```

Include dependency graph for sdhserial.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDHSerial](#)

*The class to communicate with a SDH via RS232.*

## 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](#).*

## 11.84 sdh/sdhserial.h File Reference

### 11.84.1 Detailed Description

Interface of class [SDH::cSDHSerial](#).

### 11.84.2 General file information

#### Author:

Dirk Osswald

#### Date:

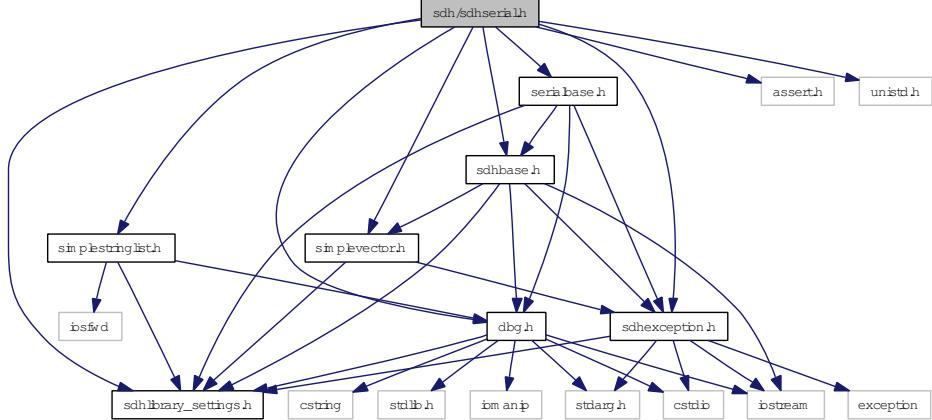
2007-02-19

### 11.84.3 Copyright

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

```
#include "sdhlibrary_settings.h"
#include <assert.h>
#include <unistd.h>
#include "dbg.h"
#include "sdhexception.h"
#include "simplevector.h"
#include "simplestringlist.h"
#include "sdhbbase.h"
#include "serialbase.h"
```

Include dependency graph for sdhserial.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSDHSerial](#)

*The class to communicate with a SDH via RS232.*

## 11.85 cpp.desire.final/sdh/serialbase.cpp File Reference

### **11.85.1 Detailed Description**

Implementation of class `SDH::cSerialBase`, a virtual base class to access serial interfaces like RS232 or CAN.

### **11.85.2 General file information**

**Author:**

Dirk Osswald

Date:

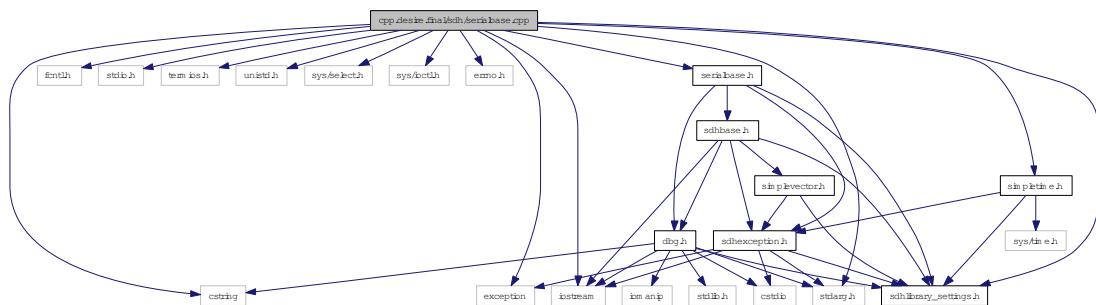
2007-02-20

### 11.85.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## 11.86 sdh/serialbase.cpp File Reference

### 11.86.1 Detailed Description

Implementation of class [SDH::cSerialBase](#), a virtual base class to access serial interfaces like RS232 or CAN.

### 11.86.2 General file information

#### Author:

Dirk Osswald

#### Date:

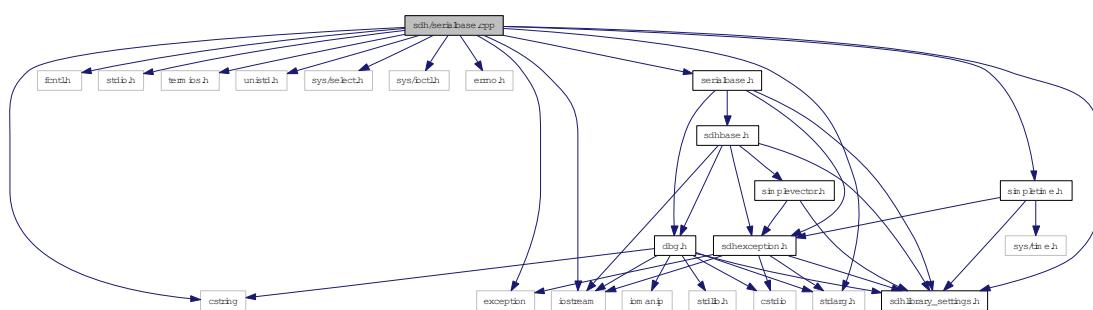
2007-02-20

### 11.86.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## 11.87 cpp.desire.final/sdh/serialbase.h File Reference

### 11.87.1 Detailed Description

Interface of class [SDH::cSerialBase](#), a virtual base class to access serial communication channels like RS232 or CAN.

### 11.87.2 General file information

#### Author:

Dirk Osswald

#### Date:

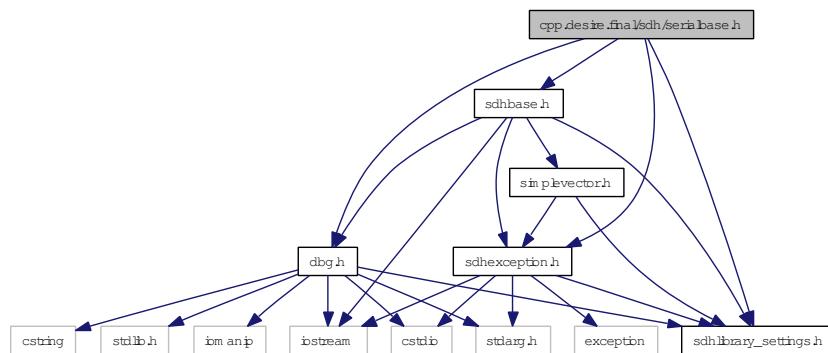
2008-05-02

### 11.87.3 Copyright

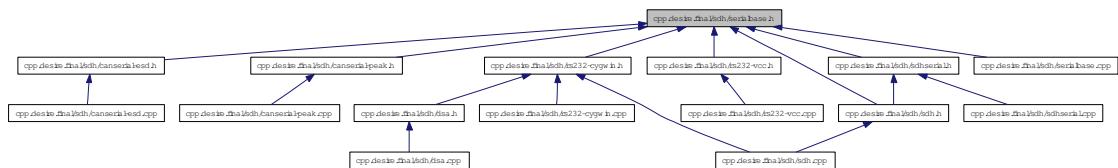
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include "sdhexception.h"
#include "dbg.h"
#include "sdhbbase.h"
```

Include dependency graph for serialbase.h:



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



## Namespaces

- namespace [SDH](#)

## 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.*

## 11.88 sdh/serialbase.h File Reference

### 11.88.1 Detailed Description

Interface of class [SDH::cSerialBase](#), a virtual base class to access serial communication channels like RS232 or CAN.

### 11.88.2 General file information

**Author:**

Dirk Osswald

**Date:**

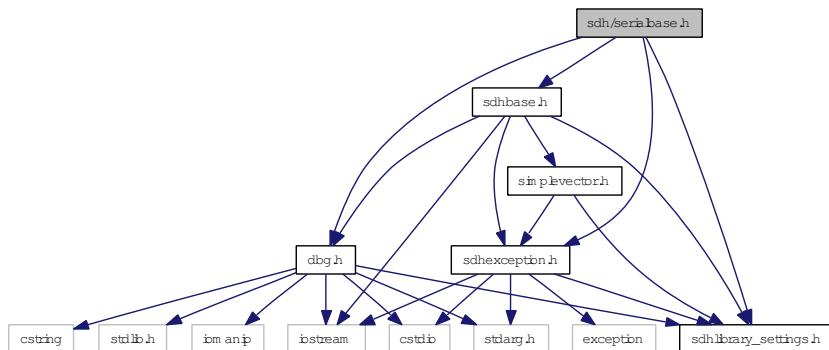
2008-05-02

### 11.88.3 Copyright

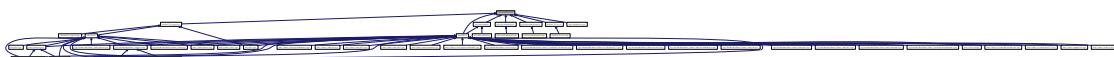
Copyright (c) 2008 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.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:



## Namespaces

- namespace [SDH](#)

## 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.*

## 11.89 cpp.desire.final/sdh/simplestringlist.cpp File Reference

### 11.89.1 Detailed Description

Implementation of class [SDH::cSimpleStringList](#).

### 11.89.2 General file information

#### Author:

Dirk Osswald

#### Date:

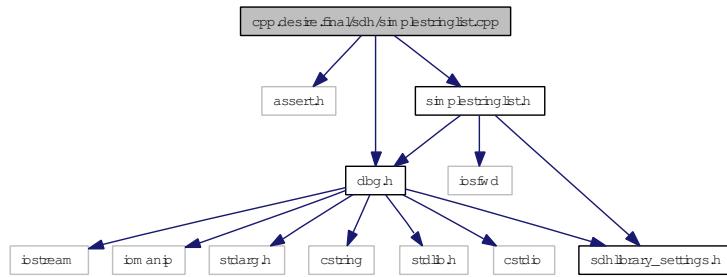
2007-02-19

### 11.89.3 Copyright

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

```
#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.*

## 11.90 **sdh/simplestringlist.cpp** File Reference

### 11.90.1 Detailed Description

Implementation of class **SDH::cSimpleStringList**.

### 11.90.2 General file information

#### Author:

Dirk Osswald

#### Date:

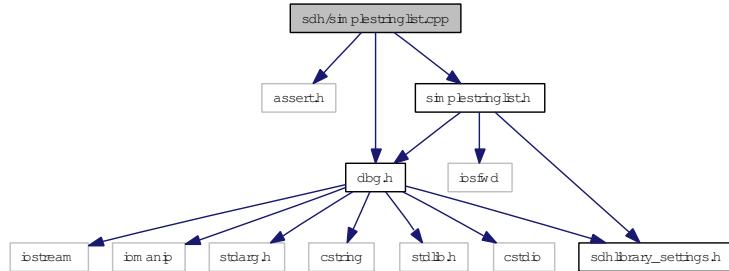
2007-02-19

### 11.90.3 Copyright

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

```
#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.*

## 11.91 cpp.desire.final/sdh/simplestringlist.h File Reference

### 11.91.1 Detailed Description

Interface of class [SDH::cSimpleStringList](#).

### 11.91.2 General file information

#### Author:

Dirk Osswald

#### Date:

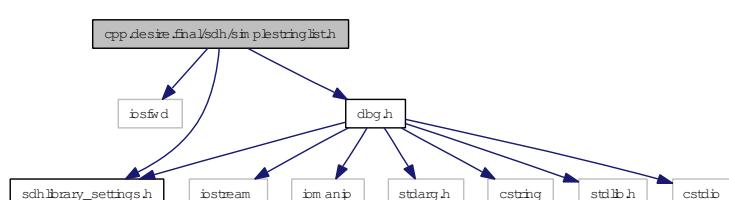
2007-02-19

### 11.91.3 Copyright

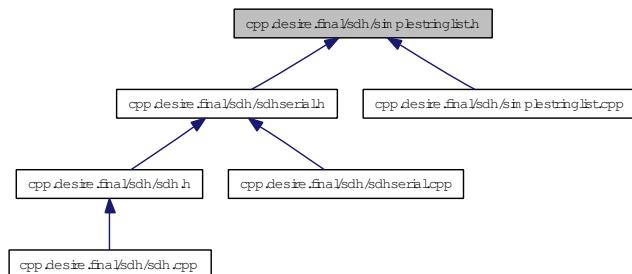
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include <iostream>
#include "dbg.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for simplestringlist.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSimpleStringList](#)

*A simple string list. (Fixed maximum number of strings of fixed maximum length).*

## Functions

- std::ostream & [SDH::operator<<](#) (std::ostream &stream, [cSimpleStringList](#) const &ssl)

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

## 11.92 sdh/simplestringlist.h File Reference

### 11.92.1 Detailed Description

Interface of class [SDH::cSimpleStringList](#).

### 11.92.2 General file information

**Author:**

Dirk Osswald

**Date:**

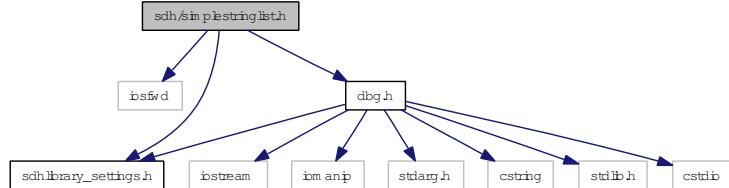
2007-02-19

### 11.92.3 Copyright

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

```
#include <iostream>
#include "dbg.h"
#include "sdhlibrary_settings.h"
```

Include dependency graph for simplestringlist.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSimpleStringList](#)

*A simple string list. (Fixed maximum number of strings of fixed maximum length).*

## Functions

- std::ostream & **SDH::operator<<** (std::ostream &stream, [cSimpleStringList](#) const &ssl)

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

## 11.93 cpp.desire.final/sdh/simpletime.h File Reference

### 11.93.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

### 11.93.2 General file information

#### Author:

Dirk Osswald

#### Date:

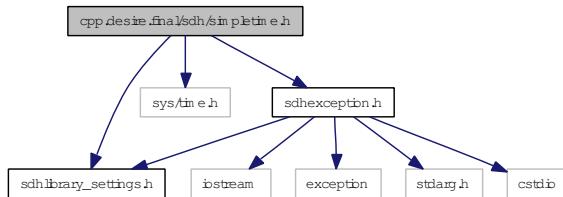
2007-02-19

### 11.93.3 Copyright

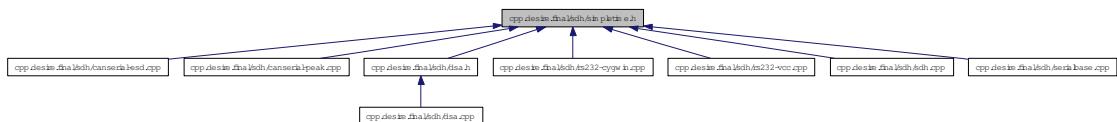
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSimpleTime](#)

*Very simple class to measure elapsed time.*

## 11.94 sdh/simpletime.h File Reference

### 11.94.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

### 11.94.2 General file information

**Author:**

Dirk Osswald

**Date:**

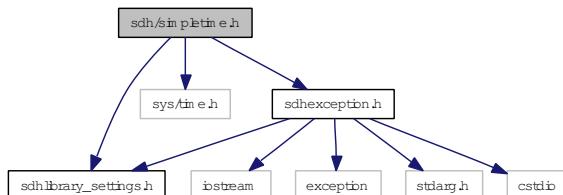
2007-02-19

### 11.94.3 Copyright

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

```
#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:



## Namespaces

- namespace **SDH**

## Classes

- class **SDH::cSimpleTime**

*Very simple class to measure elapsed time.*

## 11.95 cpp.desire.final/sdh/simplevector.cpp File Reference

### 11.95.1 Detailed Description

Implementation of class `SDH::cSimpleVector`.

### 11.95.2 General file information

#### Author:

Dirk Osswald

#### Date:

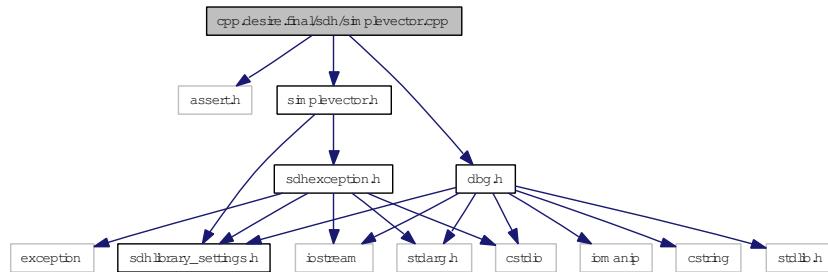
2007-02-19

### 11.95.3 Copyright

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

```
#include <assert.h>
#include "dbg.h"
#include "simplevector.h"
```

Include dependency graph for simplevector.cpp:



## 11.96 `sdh/simplevector.cpp` File Reference

### 11.96.1 Detailed Description

Implementation of class `SDH::cSimpleVector`.

### 11.96.2 General file information

#### Author:

Dirk Osswald

#### Date:

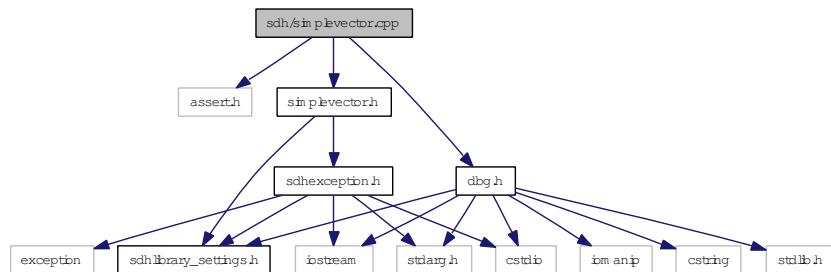
2007-02-19

### 11.96.3 Copyright

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

```
#include <assert.h>
#include "dbg.h"
#include "simplevector.h"
```

Include dependency graph for simplevector.cpp:



## 11.97 cpp.desire.final/sdh/simplevector.h File Reference

### 11.97.1 Detailed Description

Interface of class [SDH::cSimpleVector](#).

### 11.97.2 General file information

**Author:**

Dirk Osswald

**Date:**

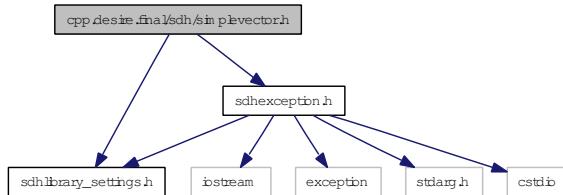
2007-02-19

### 11.97.3 Copyright

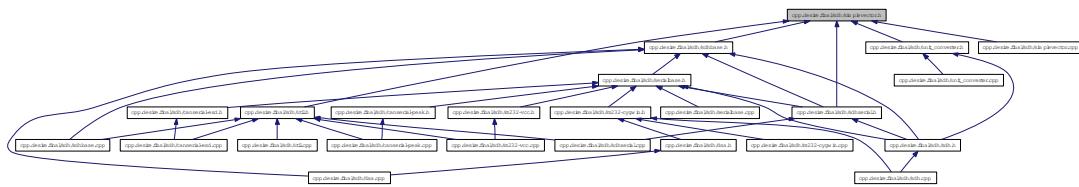
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#include "sdhlibrary_settings.h"
#include "sdhexception.h"
```

Include dependency graph for simplevector.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSimpleVectorException](#)

*Derived exception class for low-level simple vector related exceptions.*

- class [SDH::cSimpleVector](#)

*A simple vector implementation.*

## 11.98 `sdh/simplevector.h` File Reference

### 11.98.1 Detailed Description

Interface of class [SDH::cSimpleVector](#).

### 11.98.2 General file information

#### Author:

Dirk Osswald

#### Date:

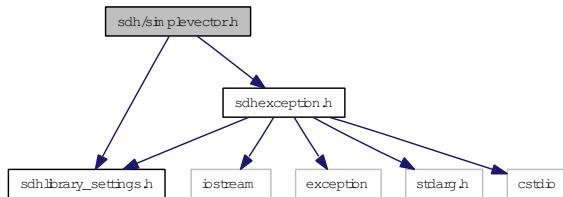
2007-02-19

### 11.98.3 Copyright

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

```
#include "sdhlibrary_settings.h"
#include "sdhexception.h"
```

Include dependency graph for simplevector.h:



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



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cSimpleVectorException](#)  
*Derived exception class for low-level simple vector related exceptions.*
- class [SDH::cSimpleVector](#)  
*A simple vector implementation.*

## 11.99 cpp.desire.final/sdh/unit\_converter.cpp File Reference

### 11.99.1 Detailed Description

Implementation of class [SDH::cUnitConverter](#).

### 11.99.2 General file information

#### Author:

Dirk Osswald

#### Date:

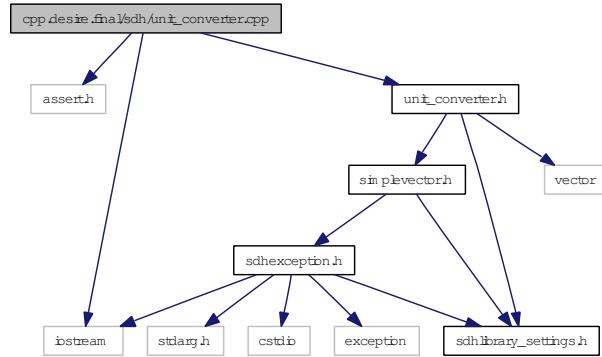
2007-02-19

### 11.99.3 Copyright

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

```
#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).*

## 11.100 `sdh/unit_converter.cpp` File Reference

### 11.100.1 Detailed Description

Implementation of class `SDH::cUnitConverter`.

### 11.100.2 General file information

#### Author:

Dirk Osswald

#### Date:

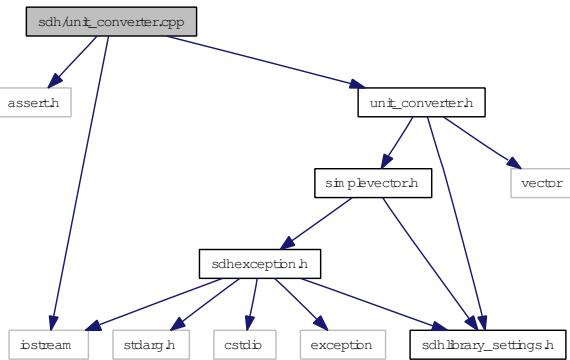
2007-02-19

### 11.100.3 Copyright

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

```
#include <assert.h>
#include <iostream>
#include "unit_converter.h"
```

Include dependency graph for `unit_converter.cpp`:



## Namespaces

- namespace `SDH`

## Functions

- `cUnitConverter const SDH::uc_identity ("any","any","?", 1.0, 0.0, 4)`

## 11.101 cpp.desire.final/sdh/unit\_converter.h File Reference

### 11.101.1 Detailed Description

Interface of class [SDH::cUnitConverter](#).

### 11.101.2 General file information

**Author:**

Dirk Osswald

**Date:**

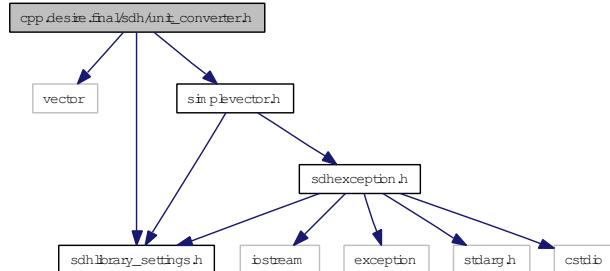
2007-02-19

### 11.101.3 Copyright

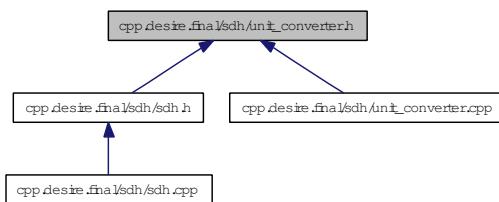
- Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cUnitConverter](#)

*Unit conversion class to convert values between physical unit systems.*

## Typedefs

- typedef double(cUnitConverter::\* [SDH::pDoubleUnitConverterFunction](#) )(double) const  
*Type of a pointer to a function like 'double cUnitConverter::ToExternal( double )' or 'cUnitConverterToInternal( double )'.*

## 11.102 `sdh/unit_converter.h` File Reference

### 11.102.1 Detailed Description

Interface of class [SDH::cUnitConverter](#).

### 11.102.2 General file information

#### Author:

Dirk Osswald

#### Date:

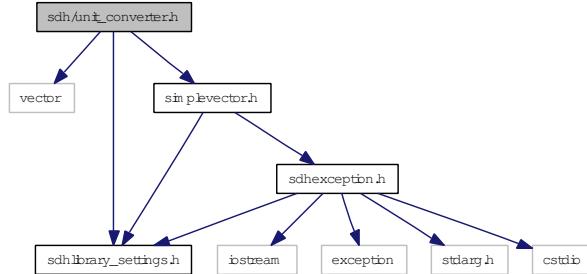
2007-02-19

### 11.102.3 Copyright

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

```
#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:



## Namespaces

- namespace [SDH](#)

## Classes

- class [SDH::cUnitConverter](#)

*Unit conversion class to convert values between physical unit systems.*

## 11.103 cpp.desire.final/sdh/util.cpp File Reference

### 11.103.1 Detailed Description

Implementation of auxilliary utility functions for SDHLIBRARY-CPP.

### 11.103.2 General file information

#### Author:

Dirk Osswald

#### Date:

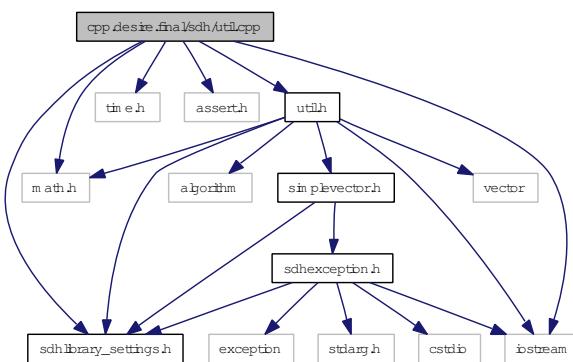
2007-02-19

### 11.103.3 Copyright

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

```
#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](#)

## 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*

### Variables

- double `SDH::M_PI` = 4.0\*atan(1.0)

## 11.104 sdh/util.cpp File Reference

### 11.104.1 Detailed Description

Implementation of auxilliary utility functions for SDHLIBRARY-CPP.

### 11.104.2 General file information

#### Author:

Dirk Osswald

#### Date:

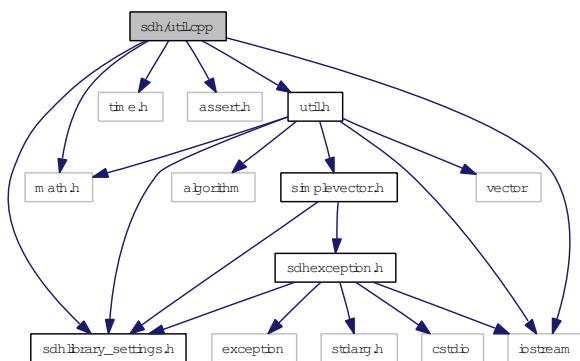
2007-02-19

### 11.104.3 Copyright

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

```
#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*

## 11.104.4 Define Documentation

### 11.104.4.1 #define \_USE\_MATH\_DEFINES

## 11.105 cpp.desire.final/sdh/util.h File Reference

### 11.105.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

### 11.105.2 General file information

#### Author:

Dirk Osswald

#### Date:

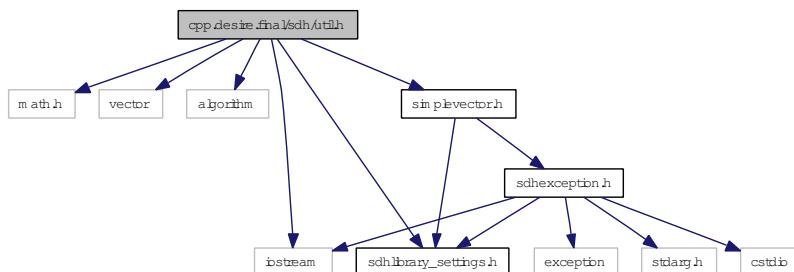
2007-02-19

### 11.105.3 Copyright

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

```
#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:



## 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*

## 11.105.4 Define Documentation

### 11.105.4.1 #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](#)

**11.105.4.2 #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](#)

## 11.106 `sdh/util.h` File Reference

### 11.106.1 Detailed Description

Interface of auxilliary utility functions for SDHLibrary-CPP.

### 11.106.2 General file information

**Author:**

Dirk Osswald

**Date:**

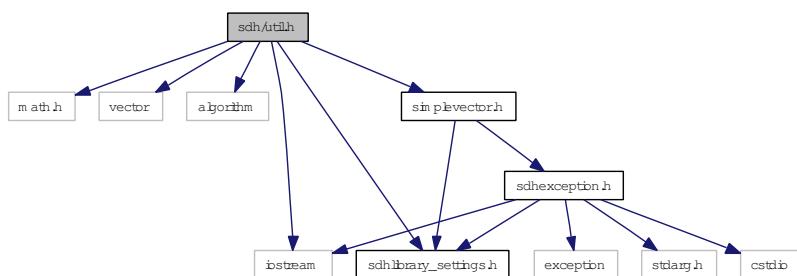
2007-02-19

### 11.106.3 Copyright

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

```
#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:



## Namespaces

- namespace [SDH](#)

## Defines

### Auxiliary functions

- #define **DEFINE\_TO\_CASECOMMAND(\_c)** case \_c: return (#\_c)
- #define **DEFINE\_TO\_CASECOMMAND\_MSG(\_c,...)** case \_c: return (#\_c ":" \_\_VA\_ARGS\_\_)

## Functions

### 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)
- 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*

## 11.106.4 Define Documentation

### 11.106.4.1 #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](#)

**11.106.4.2 #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](#)

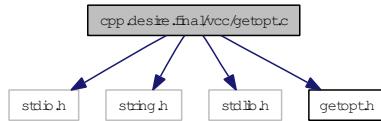
## 11.107 cpp.desire.final/sdhlibrary\_cpp.dox File Reference

## **11.108 sdhlibrary\_cpp.dox File Reference**

## 11.109 cpp.desire.final/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 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

### 11.109.1 Define Documentation

11.109.1.1 `#define _(msgid) (msgid)`

11.109.1.2 `#define GETOPT_INTERFACE_VERSION 2`

11.109.1.3 `#define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')`

11.109.1.4 `#define SWAP_FLAGS(ch1, ch2)`

### 11.109.2 Enumeration Type Documentation

11.109.2.1 anonymous enum

Enumerator:

*REQUIRE\_ORDER*

*PERMUTE*

*RETURN\_IN\_ORDER*

### 11.109.3 Function Documentation

- 11.109.3.1 **static const char\* \_\_ getopt\_initialize (int argc, char \*const \* argv, const char \* optstring) [static]**
- 11.109.3.2 **int \_\_ getopt\_internal (int argc, char \*const \* argv, const char \* optstring, const struct option \* longopts, int \* longind, int long\_only)**
- 11.109.3.3 **static void exchange (char \*\* argv) [static]**
- 11.109.3.4 **int getopt (int argc, char \*const \* argv, const char \* optstring)**
- 11.109.3.5 **static char\* my\_index (const char \* str, int chr) [static]**

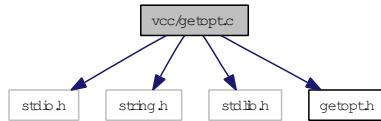
### 11.109.4 Variable Documentation

- 11.109.4.1 **int \_\_ getopt\_initialized = 0**
- 11.109.4.2 **int first\_nonopt [static]**
- 11.109.4.3 **int last\_nonopt [static]**
- 11.109.4.4 **char\* nextchar [static]**
- 11.109.4.5 **char\* optarg = NULL**
- 11.109.4.6 **int opterr = 1**
- 11.109.4.7 **int optind = 1**
- 11.109.4.8 **int optopt = '?**
- 11.109.4.9 **enum { ... } ordering [static]**
- 11.109.4.10 **char\* posixly\_correct [static]**

## 11.110 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 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

### 11.110.1 Define Documentation

11.110.1.1 `#define _(msgid) (msgid)`

11.110.1.2 `#define GETOPT_INTERFACE_VERSION 2`

11.110.1.3 `#define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')`

11.110.1.4 `#define SWAP_FLAGS(ch1, ch2)`

### 11.110.2 Enumeration Type Documentation

11.110.2.1 anonymous enum

Enumerator:

*REQUIRE\_ORDER*

*PERMUTE*

*RETURN\_IN\_ORDER*

### 11.110.3 Function Documentation

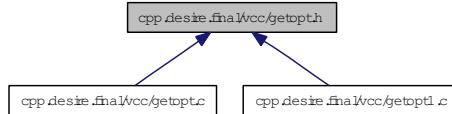
- 11.110.3.1 **static const char\* \_\_getopt\_initialize (int argc, char \*const \* argv, const char \* optstring) [static]**
- 11.110.3.2 **int \_\_getopt\_internal (int argc, char \*const \* argv, const char \* optstring, const struct option \* longopts, int \* longind, int long\_only)**
- 11.110.3.3 **static void exchange (char \*\* argv) [static]**
- 11.110.3.4 **int getopt (int argc, char \*const \* argv, const char \* optstring)**
- 11.110.3.5 **static char\* my\_index (const char \* str, int chr) [static]**

### 11.110.4 Variable Documentation

- 11.110.4.1 **int \_\_getopt\_initialized = 0**
- 11.110.4.2 **int first\_nonopt [static]**
- 11.110.4.3 **int last\_nonopt [static]**
- 11.110.4.4 **char\* nextchar [static]**
- 11.110.4.5 **char\* optarg = NULL**
- 11.110.4.6 **int opterr = 1**
- 11.110.4.7 **int optind = 1**
- 11.110.4.8 **int optopt = '?**
- 11.110.4.9 **enum { ... } ordering [static]**
- 11.110.4.10 **char\* posixly\_correct [static]**

## 11.111 cpp.desire.final/vcc/getopt.h File Reference

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



### Classes

- struct [option](#)

### Defines

- #define [\\_GETOPT\\_H](#) 1
- #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](#)

### 11.111.1 Define Documentation

11.111.1.1 `#define _GETOPT_H 1`

11.111.1.2 `#define no_argument 0`

11.111.1.3 `#define optional_argument 2`

11.111.1.4 `#define required_argument 1`

### 11.111.2 Function Documentation

11.111.2.1 `int _getopt_internal (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind, int long_only)`

11.111.2.2 `int getopt ()`

11.111.2.3 `int getopt_long (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind)`

11.111.2.4 `int getopt_long_only (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind)`

### 11.111.3 Variable Documentation

11.111.3.1 `char* optarg`

11.111.3.2 `int opterr`

11.111.3.3 `int optind`

11.111.3.4 `int optopt`

## 11.112 vcc/getopt.h File Reference

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



### Classes

- struct [option](#)

### Defines

- #define [\\_GETOPT\\_H](#) 1
- #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](#)

### 11.112.1 Define Documentation

11.112.1.1 `#define _GETOPT_H 1`

11.112.1.2 `#define no_argument 0`

11.112.1.3 `#define optional_argument 2`

11.112.1.4 `#define required_argument 1`

### 11.112.2 Function Documentation

11.112.2.1 `int _getopt_internal (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind, int long_only)`

11.112.2.2 `int getopt ()`

11.112.2.3 `int getopt_long (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind)`

11.112.2.4 `int getopt_long_only (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *longind)`

### 11.112.3 Variable Documentation

11.112.3.1 `char* optarg`

11.112.3.2 `int opterr`

11.112.3.3 `int optind`

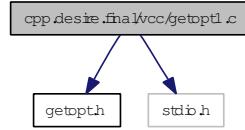
11.112.3.4 `int optopt`

## 11.113 cpp.desire.final/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)

#### 11.113.1 Define Documentation

##### 11.113.1.1 #define GETOPT\_INTERFACE\_VERSION 2

##### 11.113.1.2 #define NULL 0

#### 11.113.2 Function Documentation

##### 11.113.2.1 int getopt\_long (int *argc*, char \*const \**argv*, const char \**options*, const struct `option` \**long\_options*, int \**opt\_index*)

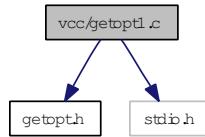
##### 11.113.2.2 int getopt\_long\_only (int *argc*, char \*const \**argv*, const char \**options*, const struct `option` \**long\_options*, int \**opt\_index*)

## 11.114 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)

#### 11.114.1 Define Documentation

##### 11.114.1.1 #define GETOPT\_INTERFACE\_VERSION 2

##### 11.114.1.2 #define NULL 0

#### 11.114.2 Function Documentation

##### 11.114.2.1 int `getopt_long` (int *argc*, char \*const \**argv*, const char \**options*, const struct `option` \**long\_options*, int \**opt\_index*)

##### 11.114.2.2 int `getopt_long_only` (int *argc*, char \*const \**argv*, const char \**options*, const struct `option` \**long\_options*, int \**opt\_index*)

## 11.115 demo/demo-griphand.cpp File Reference

### 11.115.1 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.

### 11.115.2 General file information

#### Author:

Dirk Osswald

#### Date:

2009-12-04

#### Warning:

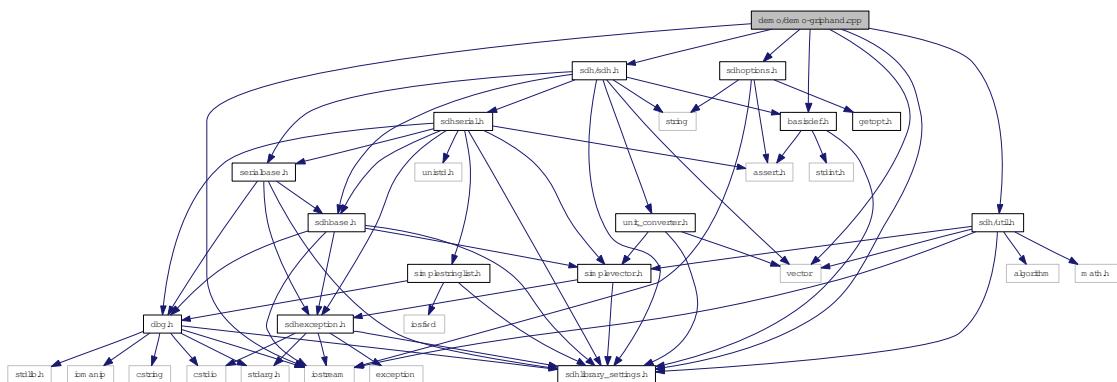
The cSDH::GripHand() function is somewhat problematic (not interruptible), see its documentation.

### 11.115.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

```
#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`

### Some informative variables

- char const \* `_help_`
- char const \* `_author_` = "Dirk Osswald: dirk.osswald@de.schunk.com"
- char const \* `_url_` = "http://www.schunk.com"
- char const \* `_version_` = "\$Id: demo-griphand.cpp 5022 2009-12-04 16:05:53Z Osswald2 \$"
- char const \* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

## 11.115.4 Function Documentation

### 11.115.4.1 int main (int *argc*, char \*\* *argv*)

## 11.115.5 Variable Documentation

**11.115.5.1 char const\* `_author_` = "Dirk Osswald: dirk.osswald@de.schunk.com"**

**11.115.5.2 char const\* `_copyright_` = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"**

**11.115.5.3 char const\* `_help_`**

#### Initial value:

```
"Demonstrate the use of the GripHand command.\n"
"(C++ demo application using the SDHLibrary-CPP library.)\n"
"\n"
"- Example usage:\n"
"- - Make SDH connected to port 2 = COM3 move:\n"
"  > demo-griphand -p 2\n"
"\n"
"- Make SDH connected to USB to RS232 converter 0 move:\n"
"  > demo-griphand --sdh_rs_device=/dev/ttyUSB0\n"
"\n"
- Get the version info of an SDH connected to port 2 = COM3\n"
  > demo-griphand --port=2 -v\n"
```

**11.115.5.4 char const\* `_url_` = "http://www.schunk.com"**

**11.115.5.5 char const\* `_version_` = "\$Id: demo-griphand.cpp 5022 2009-12-04 16:05:53Z
Osswald2 \$"**

**11.115.5.6 char const\* `usage`**

#### Initial value:

```
"usage: demo-griphand [options]\n"
```

## **11.116 doc/onlinehelp-demo-contact-grasping.exe.dox File Reference**

### **11.116.1 Detailed Description**

## **11.117 doc/onlinehelp-demo-dsa.exe.dox File Reference**

### **11.117.1 Detailed Description**

## 11.118 doc/onlinehelp-demo-dsa.exe.stackdump.dox File Reference

## **11.119 doc/onlinehelp-demo-GetAxisActualAngle.exe.dox File Reference**

### **11.119.1 Detailed Description**

## 11.120 doc/onlinehelp-demo-GetFingerXYZ.exe.dox File Reference

### 11.120.1 Detailed Description

## **11.121 doc/onlinehelp-demo-griphand.exe.dox File Reference**

### **11.121.1 Detailed Description**

## 11.122 doc/onlinehelp-demo-mimic.exe.dox File Reference

### 11.122.1 Detailed Description

**11.123 doc/onlinehelp-demo-mimic.exe.stackdump.dox File Reference**

## 11.124 doc/onlinehelp-demo-ref.exe.dox File Reference

### 11.124.1 Detailed Description

## **11.125 doc/onlinehelp-demo-simple-withtiming.exe.dox File Reference**

### **11.125.1 Detailed Description**

## 11.126 doc/onlinehelp-demo-simple.exe.dox File Reference

### 11.126.1 Detailed Description

## **11.127 doc/onlinehelp-demo-simple2.exe.dox File Reference**

### **11.127.1 Detailed Description**

## 11.128 doc/onlinehelp-demo-simple3.exe.dox File Reference

### 11.128.1 Detailed Description

## **11.129 doc/onlinehelp-demo-temperature.exe.dox File Reference**

### **11.129.1 Detailed Description**

## 11.130 doc/onlinehelp-demo-test.exe.dox File Reference

### 11.130.1 Detailed Description

## **11.131 doc/onlinehelp-demo-velocity-acceleration.exe.dox File Reference**

### **11.131.1 Detailed Description**

## 11.132 Doxyfile File Reference

### 11.132.1 Detailed Description

Doxyfile for generating documentation for SDHLibrary.cpp using doxygen.

### 11.132.2 General file information

#### Author:

Dirk Osswald

#### Date:

2007-06-14

### 11.132.3 Links

- The online documentation for Doxygen can be found at  
<http://www.stack.nl/~dimitri/doxygen/>

### 11.132.4 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

## 11.133 Makefile File Reference

### 11.133.1 Detailed Description

Makefile for [SDH](#) SDHLibrary C project.

### 11.133.2 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](#).

### 11.133.3 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

### 11.133.4 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

### 11.133.5 Links

- The online documentation for gnu make can be found at <http://www.gnu.org/software/make/manual/make.html>

### 11.133.6 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

# Index

~cDBG  
    SDH::cDBG, 84

~cDSA  
    SDH::cDSA, 94

~cRS232  
    SDH::cRS232, 135

~cSDH  
    SDH::cSDH, 163, 164

~cSDHBase  
    SDH::cSDHBase, 283

~cSDHSerial  
    SDH::cSDHSerial, 311

~cSerialBase  
    SDH::cSerialBase, 337

-  
    cpp.desire.final/vcc/getopt.c, 603  
        vcc/getopt.c, 606

\_CRT\_SECURE\_NO\_WARNINGS  
    cpp.desire.final/sdh/rs232-vcc.cpp, 532  
        sdh/rs232-vcc.cpp, 534

\_GETOPT\_H  
    cpp.desire.final/vcc/getopt.h, 609  
        vcc/getopt.h, 611

\_GetFingerXYZ  
    SDH::cSDH, 166, 213

\_USE\_MATH\_DEFINES  
    sdh/sdh.cpp, 541  
        sdh/util.cpp, 593

\_author  
    cpp.desire.final/demo/cancat.cpp, 384  
        cpp.desire.final/demo/demo-contact-grasping.cpp, 388  
    cpp.desire.final/demo/demo-dsa.cpp, 394  
        cpp.desire.final/demo/demo-GetAxisActualAngle.cpp, 400  
        cpp.desire.final/demo/demo-GetFingerXYZ.cpp, 406  
    cpp.desire.final/demo/demo-mimic.cpp, 412  
        cpp.desire.final/demo/demo-simple-withtiming.cpp, 416  
        cpp.desire.final/demo/demo-simple.cpp, 420  
        cpp.desire.final/demo/demo-simple2.cpp, 424  
        cpp.desire.final/demo/demo-simple3.cpp, 428  
        cpp.desire.final/demo/demo-temperature.cpp, 432

cpp.desire.final/demo/demo-velocity-acceleration.cpp, 436

demo-griphand.cpp, 615

demo/cancat.cpp, 386

demo/demo-contact-grasping.cpp, 391

demo/demo-dsa.cpp, 397

demo/demo-GetAxisActualAngle.cpp, 403

demo/demo-GetFingerXYZ.cpp, 409

demo/demo-mimic.cpp, 414

demo/demo-simple-withtiming.cpp, 418

demo/demo-simple.cpp, 422

demo/demo-simple2.cpp, 426

demo/demo-simple3.cpp, 430

demo/demo-temperature.cpp, 434

demo/demo-velocity-acceleration.cpp, 438

\_\_copyright\_\_  
    cpp.desire.final/demo/cancat.cpp, 384  
    cpp.desire.final/demo/demo-contact-grasping.cpp, 388  
    cpp.desire.final/demo/demo-dsa.cpp, 394  
    cpp.desire.final/demo/demo-GetAxisActualAngle.cpp, 400  
    cpp.desire.final/demo/demo-GetFingerXYZ.cpp, 406  
    cpp.desire.final/demo/demo-mimic.cpp, 412  
    cpp.desire.final/demo/demo-simple-withtiming.cpp, 416  
    cpp.desire.final/demo/demo-simple.cpp, 420  
    cpp.desire.final/demo/demo-simple2.cpp, 424  
    cpp.desire.final/demo/demo-simple3.cpp, 428  
    cpp.desire.final/demo/demo-temperature.cpp, 432

cpp.desire.final/demo/demo-velocity-acceleration.cpp, 436

demo-griphand.cpp, 615

demo/cancat.cpp, 386

demo/demo-contact-grasping.cpp, 391

demo/demo-dsa.cpp, 397

demo/demo-GetAxisActualAngle.cpp, 403

demo/demo-GetFingerXYZ.cpp, 409

demo/demo-mimic.cpp, 414

demo/demo-simple-withtiming.cpp, 418

demo/demo-simple.cpp, 422

demo/demo-simple2.cpp, 426

demo/demo-simple3.cpp, 430

demo/demo-temperature.cpp, 434  
demo/demo-velocity-acceleration.cpp, 438  
\_\_getopt\_initialized  
  cpp.desire.final/vcc/getopt.c, 604  
  vcc/getopt.c, 607  
\_\_help\_\_  
  cpp.desire.final/demo/cancat.cpp, 384  
  cpp.desire.final/demo/demo-contact-  
    grasping.cpp, 388  
  cpp.desire.final/demo/demo-dsa.cpp, 394  
  cpp.desire.final/demo/demo-  
    GetAxisActualAngle.cpp, 400  
  cpp.desire.final/demo/demo-  
    GetFingerXYZ.cpp, 406  
  cpp.desire.final/demo/demo-mimic.cpp, 412  
  cpp.desire.final/demo/demo-simple-  
    withtiming.cpp, 416  
  cpp.desire.final/demo/demo-simple.cpp, 420  
  cpp.desire.final/demo/demo-simple2.cpp, 424  
  cpp.desire.final/demo/demo-simple3.cpp, 428  
  cpp.desire.final/demo/demo-temperature.cpp,  
    432  
  cpp.desire.final/demo/demo-velocity-  
    acceleration.cpp, 436  
demo-griphand.cpp, 615  
demo/cancat.cpp, 386  
demo/demo-contact-grasping.cpp, 391  
demo/demo-dsa.cpp, 397  
demo/demo-GetAxisActualAngle.cpp, 403  
demo/demo-GetFingerXYZ.cpp, 409  
demo/demo-mimic.cpp, 414  
demo/demo-simple-withtiming.cpp, 418  
demo/demo-simple.cpp, 422  
demo/demo-simple2.cpp, 426  
demo/demo-simple3.cpp, 430  
demo/demo-temperature.cpp, 434  
demo/demo-velocity-acceleration.cpp, 438  
\_\_packed\_\_  
  SDH::cDSA, 101  
\_\_url\_\_  
  cpp.desire.final/demo/cancat.cpp, 384  
  cpp.desire.final/demo/demo-contact-  
    grasping.cpp, 389  
  cpp.desire.final/demo/demo-dsa.cpp, 395  
  cpp.desire.final/demo/demo-  
    GetAxisActualAngle.cpp, 400  
  cpp.desire.final/demo/demo-  
    GetFingerXYZ.cpp, 407  
  cpp.desire.final/demo/demo-mimic.cpp, 412  
  cpp.desire.final/demo/demo-simple-  
    withtiming.cpp, 416  
  cpp.desire.final/demo/demo-simple.cpp, 420  
  cpp.desire.final/demo/demo-simple2.cpp, 424  
  cpp.desire.final/demo/demo-simple3.cpp, 428  
cpp.desire.final/demo/demo-temperature.cpp,  
  432  
  cpp.desire.final/demo/demo-velocity-  
    acceleration.cpp, 436  
  demo-griphand.cpp, 615  
  demo/cancat.cpp, 386  
  demo/demo-contact-grasping.cpp, 392  
  demo/demo-dsa.cpp, 398  
  demo/demo-GetAxisActualAngle.cpp, 403  
  demo/demo-GetFingerXYZ.cpp, 410  
  demo/demo-mimic.cpp, 414  
  demo/demo-simple-withtiming.cpp, 418  
  demo/demo-simple.cpp, 422  
  demo/demo-simple2.cpp, 426  
  demo/demo-simple3.cpp, 430  
  demo/demo-temperature.cpp, 434  
  demo/demo-velocity-acceleration.cpp, 438  
\_\_version\_\_  
  cpp.desire.final/demo/cancat.cpp, 384  
  cpp.desire.final/demo/demo-contact-  
    grasping.cpp, 389  
  cpp.desire.final/demo/demo-dsa.cpp, 395  
  cpp.desire.final/demo/demo-  
    GetAxisActualAngle.cpp, 400  
  cpp.desire.final/demo/demo-  
    GetFingerXYZ.cpp, 407  
  cpp.desire.final/demo/demo-mimic.cpp, 412  
  cpp.desire.final/demo/demo-simple-  
    withtiming.cpp, 416  
  cpp.desire.final/demo/demo-simple.cpp, 420  
  cpp.desire.final/demo/demo-simple2.cpp, 424  
  cpp.desire.final/demo/demo-simple3.cpp, 428  
  cpp.desire.final/demo/demo-temperature.cpp,  
    432  
  cpp.desire.final/demo/demo-velocity-  
    acceleration.cpp, 436  
  demo-griphand.cpp, 615  
  demo/cancat.cpp, 386  
  demo/demo-contact-grasping.cpp, 392  
  demo/demo-dsa.cpp, 398  
  demo/demo-GetAxisActualAngle.cpp, 403  
  demo/demo-GetFingerXYZ.cpp, 410  
  demo/demo-mimic.cpp, 414  
  demo/demo-simple-withtiming.cpp, 418  
  demo/demo-simple.cpp, 422  
  demo/demo-simple2.cpp, 426  
  demo/demo-simple3.cpp, 430  
  demo/demo-temperature.cpp, 434  
  demo/demo-velocity-acceleration.cpp, 438  
\_\_getopt\_initialize  
  cpp.desire.final/vcc/getopt.c, 604  
  vcc/getopt.c, 607  
\_\_getopt\_internal  
  cpp.desire.final/vcc/getopt.c, 604

cpp.desire.final/vcc/getopt.h, 609  
 vcc/getopt.c, 607  
 vcc/getopt.h, 611

a

- SDH::cSDHSerial, 316, 326
- a\_time
  - SDH::cSimpleTime, 351
- active\_interface
  - SDH::cDSA::sControllerInfo, 105
- AddByte
  - SDH::cCRC, 78
- alim
  - SDH::cSDHSerial, 315, 325
- All
  - SDH::cSDHBase, 276, 280
- all\_axes
  - SDH::cSDH, 263, 266
- all\_axes\_used
  - SDH::cSDHBase, 288
- all\_fingers
  - SDH::cSDH, 263, 266
- all\_real\_axes
  - SDH::cSDH, 263, 266
- all\_temperature\_sensors
  - SDH::cSDH, 263, 266
- apply
  - SDH, 48
- Approx
  - SDH, 48
- architecture.dox, 369
- area
  - SDH::cDSA::sContactInfo, 104
- AxisAnglesToFingerAngles
  - cpp.desire.final/demo/demo-contact-grasping.cpp, 388
  - demo/demo-contact-grasping.cpp, 391
- AxisCommand
  - SDH::cSDHSerial, 313, 323
- baudrate
  - SDH::cCANSerial\_ESD, 60
  - SDH::cCANSerial\_PEAK, 72
  - SDH::cRS232, 140
- BaudrateToBaudrateCode
  - SDH::cCANSerial\_ESD, 58, 59
  - SDH::cCANSerial\_PEAK, 69, 70
  - SDH::cRS232, 136, 138
- c\_str
  - SDH::cMsg, 130
- calib\_pressure
  - SDH::cDSA, 102
- calib\_voltage
- SDH::cDSA, 102
- can\_baudrate
- cSDHOptions, 304
- SDH::cDSA::sControllerInfo, 105
- CAN\_ESD\_RXQUEUESIZE
  - cpp.desire.final/sdh/canserial-esd.h, 472
  - sdh/canserial-esd.h, 474
- CAN\_ESD\_TXQUEUESIZE
  - cpp.desire.final/sdh/canserial-esd.h, 472
  - sdh/canserial-esd.h, 474
- can\_id
  - SDH::cDSA::sControllerInfo, 105
- cCANSerial\_ESD
  - SDH::cCANSerial\_ESD, 57, 58
- cCANSerial\_ESDEception
  - SDH::cCANSerial\_ESDEception, 64
- cCANSerial\_PEAK
  - SDH::cCANSerial\_PEAK, 68, 69
- cCANSerial\_PEAKException
  - SDH::cCANSerial\_PEAKException, 75
- cCRC
  - SDH::cCRC, 77
- cCRC\_DSACON32m
  - SDH::cCRC\_DSACON32m, 81
- cDBG
  - SDH::cDBG, 84
- cdbg
  - cpp.desire.final/demo/demo-contact-grasping.cpp, 389
  - cpp.desire.final/demo/dsaboost.cpp, 440
  - demo/demo-contact-grasping.cpp, 392
  - demo/dsaboost.cpp, 442
  - SDH::cSDHBase, 286
- cDSA
  - SDH::cDSA, 93, 94
- cDSAException
  - SDH::cDSAException, 115
- cDSAUpdater
  - SDH::cDSAUpdater, 118
- cells\_x
  - SDH::cDSA::sMatrixInfo, 107
- cells\_y
  - SDH::cDSA::sMatrixInfo, 107
- CheckIndex
  - SDH::cSDHBase, 283, 285
- CheckRange
  - SDH::cSDHBase, 283–285
- cIsGraspedBase
  - SDH::cIsGraspedBase, 122
- cIsGraspedByArea
  - SDH::cIsGraspedByArea, 126
- Close
  - SDH::cCANSerial\_ESD, 58, 59
  - SDH::cCANSerial\_PEAK, 70, 71

SDH::cDSA, 96, 99  
SDH::cRS232, 136–139  
SDH::cSDH, 172, 220  
SDH::cSDHSerial, 312, 322  
SDH::cSerialBase, 338, 339  
cMsg  
    SDH::cMsg, 129  
cog\_x  
    SDH::cDSA::sContactInfo, 104  
cog\_y  
    SDH::cDSA::sContactInfo, 104  
com  
    SDH::cSDH, 263, 266  
    SDH::cSDHSerial, 331, 332  
comm\_interface  
    SDH::cDSA, 101  
    SDH::cSDH, 263  
CompareReleases  
    SDH, 48  
Compile time settings, 23  
con  
    SDH::cSDHSerial, 317, 327  
connectors.dox, 375  
contact\_area\_cell\_threshold  
    SDH::cDSA, 102  
contact\_force\_cell\_threshold  
    SDH::cDSA, 102  
controller\_info  
    SDH::cDSA, 101  
controller\_type\_name  
    SDH::cSDHBase, 287, 289  
controllerinfo  
    cSDHOptions, 304  
cpp.desire.final/architecture.dox, 372  
cpp.desire.final/connectors.dox, 379  
cpp.desire.final/demo/cancat.cpp, 383  
    \_\_author\_\_, 384  
    \_\_copyright\_\_, 384  
    \_\_help\_\_, 384  
    \_\_url\_\_, 384  
    \_\_version\_\_, 384  
    main, 384  
cpp.desire.final/demo/demo-contact-grasping.cpp,  
    387  
    \_\_author\_\_, 388  
    \_\_copyright\_\_, 388  
    \_\_help\_\_, 388  
    \_\_url\_\_, 389  
    \_\_version\_\_, 389  
    AxisAnglesToFingerAngles, 388  
    cdbg, 389  
    GotoStartPose, 388  
    main, 388  
    usage, 389  
cpp.desire.final/demo/demo-dsa.cpp, 393  
    \_\_author\_\_, 394  
    \_\_copyright\_\_, 394  
    \_\_help\_\_, 394  
    \_\_url\_\_, 395  
    \_\_version\_\_, 395  
    main, 394  
    usage, 395  
cpp.desire.final/demo/demo-  
    GetAxisActualAngle.cpp, 399  
    \_\_author\_\_, 400  
    \_\_copyright\_\_, 400  
    \_\_help\_\_, 400  
    \_\_url\_\_, 400  
    \_\_version\_\_, 400  
    main, 400  
    usage, 400  
cpp.desire.final/demo/demo-GetFingerXYZ.cpp,  
    405  
    \_\_author\_\_, 406  
    \_\_copyright\_\_, 406  
    \_\_help\_\_, 406  
    \_\_url\_\_, 407  
    \_\_version\_\_, 407  
    main, 406  
    usage, 407  
cpp.desire.final/demo/demo-mimic.cpp, 411  
    \_\_author\_\_, 412  
    \_\_copyright\_\_, 412  
    \_\_help\_\_, 412  
    \_\_url\_\_, 412  
    \_\_version\_\_, 412  
    GetHand, 412  
    main, 412  
cpp.desire.final/demo/demo-simple-  
    withtiming.cpp, 415  
    \_\_author\_\_, 416  
    \_\_copyright\_\_, 416  
    \_\_help\_\_, 416  
    \_\_url\_\_, 416  
    \_\_version\_\_, 416  
    main, 416  
    usage, 416  
cpp.desire.final/demo/demo-simple.cpp, 419  
    \_\_author\_\_, 420  
    \_\_copyright\_\_, 420  
    \_\_help\_\_, 420  
    \_\_url\_\_, 420  
    \_\_version\_\_, 420  
    main, 420  
    usage, 420  
cpp.desire.final/demo/demo-simple2.cpp, 423  
    \_\_author\_\_, 424  
    \_\_copyright\_\_, 424

\_\_help\_\_, 424  
 \_\_url\_\_, 424  
 \_\_version\_\_, 424  
 main, 424  
 cpp.desire.final/demo/demo-simple3.cpp, 427  
   \_\_author\_\_, 428  
   \_\_copyright\_\_, 428  
   \_\_help\_\_, 428  
   \_\_url\_\_, 428  
   \_\_version\_\_, 428  
   main, 428  
 cpp.desire.final/demo/demo-temperature.cpp, 431  
   \_\_author\_\_, 432  
   \_\_copyright\_\_, 432  
   \_\_help\_\_, 432  
   \_\_url\_\_, 432  
   \_\_version\_\_, 432  
   main, 432  
 cpp.desire.final/demo/demo-velocity-acceleration.cpp, 435  
   \_\_author\_\_, 436  
   \_\_copyright\_\_, 436  
   \_\_help\_\_, 436  
   \_\_url\_\_, 436  
   \_\_version\_\_, 436  
   main, 436  
   usage, 436  
 cpp.desire.final/demo/dsaboostrap.cpp, 439  
   cdbg, 440  
 cpp.desire.final/demo/dsaboostrap.h, 443  
 cpp.desire.final/demo/sdhoptions.cpp, 447  
   sdhoptions\_long\_options, 448  
   sdhoptions\_short\_options, 449  
   sdhusage\_dsaicom, 449  
   sdhusage\_dsaother, 449  
   sdhusage\_general, 450  
   sdhusage\_sdhcom\_cancommon, 450  
   sdhusage\_sdhcom\_common, 450  
   sdhusage\_sdhcom\_esdcan, 451  
   sdhusage\_sdhcom\_peakcan, 451  
   sdhusage\_sdhcom\_serial, 451  
   sdhusage\_sdhoother, 451  
 cpp.desire.final/demo/sdhoptions.h, 459  
   SDHUSAGE\_DEFAULT, 460  
 cpp.desire.final/sdh/basisdef.h, 463  
   SDH\_ASSERT\_TYPESIZES, 464  
 cpp.desire.final/sdh/canserial-esd.cpp, 467  
   ESD\_strerror, 468  
 cpp.desire.final/sdh/canserial-esd.h, 471  
   CAN\_ESD\_RXQUEUESIZE, 472  
   CAN\_ESD\_TXQUEUESIZE, 472  
   NOMINMAX, 472  
 cpp.desire.final/sdh/canserial-peak.cpp, 475  
   PEAK\_strerror, 476  
     USE\_HANDLE, 476  
     USE\_HANDLES, 476  
 cpp.desire.final/sdh/canserial-peak.h, 479  
   M\_CMSG\_MSG, 480  
 cpp.desire.final/sdh/crc.cpp, 483  
 cpp.desire.final/sdh/crc.h, 485  
 cpp.desire.final/sdh/dbg.h, 489  
   V, 490  
   VAR, 490  
 cpp.desire.final/sdh/dsa.cpp, 493  
   PRINT\_MEMBER, 494  
   PRINT\_MEMBER\_HEX, 494  
   SDH\_NAMESPACE\_PREFIX, 494  
 cpp.desire.final/sdh/dsa.h, 497  
   DSA\_MAX\_PREAMBLE\_SEARCH, 499  
 cpp.desire.final/sdh/release.h, 503  
   PROJECT\_COPYRIGHT, 504  
   PROJECT\_DATE, 504  
   PROJECT\_NAME, 504  
   PROJECT\_RELEASE, 504  
 cpp.desire.final/sdh/rs232-cygwin.cpp, 523  
   DBG, 524  
   SDH\_RS232\_CYGWIN\_DEBUG, 524  
   SDH\_RS232\_CYGWIN\_DEBUG\_ASCII, 524  
   StrDupNew, 524  
 cpp.desire.final/sdh/rs232-cygwin.h, 527  
 cpp.desire.final/sdh/rs232-vcc.cpp, 531  
   \_CRT\_SECURE\_NO\_WARNINGS, 532  
   DBG, 532  
   GetLastErrorMessage, 532  
   SDH\_RS232\_VCC\_DEBUG, 532  
   SDH\_RS232\_VCC\_DEBUG\_ASCII, 532  
 cpp.desire.final/sdh/rs232-vcc.h, 535  
   SDH\_RS232\_VCC\_ASYNC, 536  
 cpp.desire.final/sdh/sdh.cpp, 539  
 cpp.desire.final/sdh/sdh.h, 542  
 cpp.desire.final/sdh/sdhbase.cpp, 546  
 cpp.desire.final/sdh/sdhbase.h, 548  
 cpp.desire.final/sdh/sdhexception.cpp, 552  
 cpp.desire.final/sdh/sdhexception.h, 554  
 cpp.desire.final/sdh/sdhlibrary\_settings.h, 558  
 cpp.desire.final/sdh/sdhserial.cpp, 560  
 cpp.desire.final/sdh/sdhserial.h, 562  
 cpp.desire.final/sdh/serialbase.cpp, 566  
 cpp.desire.final/sdh/serialbase.h, 568  
 cpp.desire.final/sdh/simplestringlist.cpp, 572  
 cpp.desire.final/sdh/simplestringlist.h, 574  
 cpp.desire.final/sdh/simpletime.h, 578  
 cpp.desire.final/sdh/simplevector.cpp, 580  
 cpp.desire.final/sdh/simplevector.h, 582  
 cpp.desire.final/sdh/unit\_converter.cpp, 585  
 cpp.desire.final/sdh/unit\_converter.h, 587  
 cpp.desire.final/sdh/util.cpp, 590

cpp.desire.final/sdh/util.h, 594  
    DEFINE\_TO\_CASECOMMAND, 595  
    DEFINE\_TO\_CASECOMMAND\_MSG, 595  
cpp.desire.final/sdhlibrary\_cpp.dox, 600  
cpp.desire.final/vcc/getopt.c, 602  
    -, 603  
    \_getopt\_initialized, 604  
    \_getopt\_initialize, 604  
    \_getopt\_internal, 604  
    exchange, 604  
    first\_nonopt, 604  
    getopt, 604  
    GETOPT\_INTERFACE\_VERSION, 603  
    last\_nonopt, 604  
    my\_index, 604  
    nextchar, 604  
    NONOPTION\_P, 603  
    optarg, 604  
    opterr, 604  
    optind, 604  
    optopt, 604  
    ordering, 604  
    PERMUTE, 603  
    posixly\_correct, 604  
    REQUIRE\_ORDER, 603  
    RETURN\_IN\_ORDER, 603  
    SWAP\_FLAGS, 603  
cpp.desire.final/vcc/getopt.h, 608  
    \_GETOPT\_H, 609  
    \_getopt\_internal, 609  
    getopt, 609  
    getopt\_long, 609  
    getopt\_long\_only, 609  
    no\_argument, 609  
    optarg, 609  
    opterr, 609  
    optind, 609  
    optional\_argument, 609  
    optopt, 609  
    required\_argument, 609  
cpp.desire.final/vcc/getopt1.c, 612  
    GETOPT\_INTERFACE\_VERSION, 612  
    getopt\_long, 612  
    getopt\_long\_only, 612  
    NULL, 612  
crc\_table  
    SDH::cCRC, 79  
crc\_table\_dsacon32m  
    SDH::cCRC\_DSACON32m, 81  
cRS232  
    SDH::cRS232, 134, 135  
cRS232Exception  
    SDH::cRS232Exception, 143  
cSDH  
    SDH::cSDH, 161, 163  
cSDHBase  
    SDH::cSDHBase, 283  
cSDHErrorCommunication  
    SDH::cSDHErrorCommunication, 292  
cSDHErrorInvalidParameter  
    SDH::cSDHErrorInvalidParameter, 295  
cSDHLibraryException  
    SDH::cSDHLibraryException, 297, 298  
cSDHOPTIONS, 300  
    can\_baudrate, 304  
    controllerinfo, 304  
    cSDHOPTIONS, 301  
    debug\_level, 304  
    debuglog, 304  
    do\_RLE, 304  
    dsa\_rs\_device, 304  
    dsaport, 304  
    framerate, 304  
    fullframe, 304  
    id\_read, 304  
    id\_write, 304  
    matrixinfo, 304  
    MAX\_DEV\_LENGTH, 304  
    net, 304  
    Parse, 301, 302  
    period, 304  
    rs232\_baudrate, 304  
    sdh\_canpeak\_device, 304  
    sdh\_rs\_device, 304  
    sdhport, 304  
    sensorinfo, 304  
    timeout, 304  
    usage, 304  
    use\_can\_esd, 304  
    use\_can\_peak, 304  
    use\_fahrenheit, 304  
    use\_radians, 304  
cSDHSerial  
    SDH::cSDHSerial, 311  
cSerialBase  
    SDH::cSerialBase, 337  
cSerialBaseException  
    SDH::cSerialBaseException, 344  
cSimpleStringList  
    SDH::cSimpleStringList, 347  
cSimpleTime  
    SDH::cSimpleTime, 350  
cSimpleVector  
    SDH::cSimpleVector, 354  
cSimpleVectorException  
    SDH::cSimpleVectorException, 359  
cUnitConverter  
    SDH::cUnitConverter, 362

current\_crc  
     SDH::cCRC, 78

current\_line  
     SDH::cSimpleStringList, 348

CurrentLine  
     SDH::cSimpleStringList, 347

d  
     SDH::cSDH, 263

DBG  
     cpp.desire.final/sdh/rs232-cygwin.cpp, 524  
     cpp.desire.final/sdh/rs232-vcc.cpp, 532  
     sdh/rs232-cygwin.cpp, 526  
     sdh/rs232-vcc.cpp, 534

dbg  
     SDH::cDSA, 101  
     SDH::cSerialBase, 341

debug  
     SDH::cSDHSerial, 315, 325

debug\_color  
     SDH::cDBG, 86

debug\_flag  
     SDH::cDBG, 86

debug\_level  
     cSDHOptions, 304  
     SDH::cSDHBase, 286

debuglog  
     cSDHOptions, 304

decimal\_places  
     SDH::cUnitConverter, 365

DEFAULT\_ERROR\_THRESHOLD  
     SDH::cDSAUpdater, 119

DEFINE\_TO\_CASECOMMAND  
     cpp.desire.final/sdh/util.h, 595  
     sdh/util.h, 598

DEFINE\_TO\_CASECOMMAND\_MSG  
     cpp.desire.final/sdh/util.h, 595  
     sdh/util.h, 599

DegToRad  
     SDH, 49

demo  
     SDH::cSDHSerial, 314, 324

demo-griphand.cpp  
     \_\_author\_\_, 615  
     \_\_copyright\_\_, 615  
     \_\_help\_\_, 615  
     \_\_url\_\_, 615  
     \_\_version\_\_, 615  
     main, 615  
     usage, 615

demo/cancat.cpp, 385  
     \_\_author\_\_, 386  
     \_\_copyright\_\_, 386  
     \_\_help\_\_, 386

demo/demo-contact-grasping.cpp, 390  
     \_\_author\_\_, 391  
     \_\_copyright\_\_, 391  
     \_\_help\_\_, 391  
     \_\_url\_\_, 392  
     \_\_version\_\_, 392  
     AxisAnglesToFingerAngles, 391  
     cdbg, 392  
     GotoStartPose, 391  
     main, 391  
     usage, 392

demo/demo-dsa.cpp, 396  
     \_\_author\_\_, 397  
     \_\_copyright\_\_, 397  
     \_\_help\_\_, 397  
     \_\_url\_\_, 398  
     \_\_version\_\_, 398  
     main, 397  
     usage, 398

demo/demo-GetAxisActualAngle.cpp, 402  
     \_\_author\_\_, 403  
     \_\_copyright\_\_, 403  
     \_\_help\_\_, 403  
     \_\_url\_\_, 403  
     \_\_version\_\_, 403  
     main, 403  
     usage, 403

demo/demo-GetFingerXYZ.cpp, 408  
     \_\_author\_\_, 409  
     \_\_copyright\_\_, 409  
     \_\_help\_\_, 409  
     \_\_url\_\_, 410  
     \_\_version\_\_, 410  
     main, 409  
     usage, 410

demo/demo-griphand.cpp, 614

demo/demo-mimic.cpp, 413  
     \_\_author\_\_, 414  
     \_\_copyright\_\_, 414  
     \_\_help\_\_, 414  
     \_\_url\_\_, 414  
     \_\_version\_\_, 414  
     GetHand, 414  
     main, 414

demo/demo-simple-withtiming.cpp, 417  
     \_\_author\_\_, 418  
     \_\_copyright\_\_, 418  
     \_\_help\_\_, 418  
     \_\_url\_\_, 418  
     \_\_version\_\_, 418  
     main, 418

usage, 418  
demo/demo-simple.cpp, 421  
    \_\_author\_\_, 422  
    \_\_copyright\_\_, 422  
    \_\_help\_\_, 422  
    \_\_url\_\_, 422  
    \_\_version\_\_, 422  
    main, 422  
    usage, 422  
demo/demo-simple2.cpp, 425  
    \_\_author\_\_, 426  
    \_\_copyright\_\_, 426  
    \_\_help\_\_, 426  
    \_\_url\_\_, 426  
    \_\_version\_\_, 426  
    main, 426  
demo/demo-simple3.cpp, 429  
    \_\_author\_\_, 430  
    \_\_copyright\_\_, 430  
    \_\_help\_\_, 430  
    \_\_url\_\_, 430  
    \_\_version\_\_, 430  
    main, 430  
demo/demo-temperature.cpp, 433  
    \_\_author\_\_, 434  
    \_\_copyright\_\_, 434  
    \_\_help\_\_, 434  
    \_\_url\_\_, 434  
    \_\_version\_\_, 434  
    main, 434  
demo/demo-velocity-acceleration.cpp, 437  
    \_\_author\_\_, 438  
    \_\_copyright\_\_, 438  
    \_\_help\_\_, 438  
    \_\_url\_\_, 438  
    \_\_version\_\_, 438  
    main, 438  
    usage, 438  
demo/dsaboostrap.cpp, 441  
    cdbg, 442  
demo/dsaboostrap.h, 445  
demo/sdhoptions.cpp, 453  
    sdhoptions\_long\_options, 454  
    sdhoptions\_short\_options, 455  
    sdhusage\_dsacom, 455  
    sdhusage\_dsaother, 455  
    sdhusage\_general, 456  
    sdhusage\_sdhcom\_cancommon, 456  
    sdhusage\_sdhcom\_common, 456  
    sdhusage\_sdhcom\_esdcan, 457  
    sdhusage\_sdhcom\_peakcan, 457  
    sdhusage\_sdhcom\_serial, 457  
    sdhusage\_sdhother, 457  
demo/sdhoptions.h, 461  
    SDHUSAGE\_DEFAULT, 462  
Demonstration programs, 25  
Derived compile time settings, 24  
device\_format\_string  
    SDH::cRS232, 140  
do\_RLE  
    cSDHOptions, 304  
    SDH::cDSA, 101  
doc/onlinehelp-demo-contact-grasping.exe.dox, 616  
doc/onlinehelp-demo-dsa.exe.dox, 617  
doc/onlinehelp-demo-dsa.exe.stackdump.dox, 618  
doc/onlinehelp-demo-  
    GetAxisActualAngle.exe.dox, 619  
doc/onlinehelp-demo-GetFingerXYZ.exe.dox, 620  
doc/onlinehelp-demo-griphand.exe.dox, 621  
doc/onlinehelp-demo-mimic.exe.dox, 622  
doc/onlinehelp-demo-mimic.exe.stackdump.dox, 623  
doc/onlinehelp-demo-ref.exe.dox, 624  
doc/onlinehelp-demo-simple-withtiming.exe.dox, 625  
doc/onlinehelp-demo-simple.exe.dox, 626  
doc/onlinehelp-demo-simple2.exe.dox, 627  
doc/onlinehelp-demo-simple3.exe.dox, 628  
doc/onlinehelp-demo-temperature.exe.dox, 629  
doc/onlinehelp-demo-test.exe.dox, 630  
doc/onlinehelp-demo-velocity-  
    acceleration.exe.dox, 631  
Doxyfile, 632  
DSA\_MAX\_PREAMBLE\_SEARCH  
    cpp.desire.final/sdh/dsa.h, 499  
    sdh/dsa.h, 502  
dsa\_rs\_device  
    cSDHOptions, 304  
dsaport  
    cSDHOptions, 304  
eAS\_CCW\_BLOCKED  
    SDH::cSDH, 160, 161  
eAS\_CW\_BLOCKED  
    SDH::cSDH, 160, 161  
eAS\_DIMENSION  
    SDH::cSDH, 160, 161  
eAS\_DISABLED  
    SDH::cSDH, 160, 161  
eAS\_IDLE  
    SDH::cSDH, 159–161  
eAS\_LIMITS\_REACHED  
    SDH::cSDH, 160, 161  
eAS\_NOT\_INITIALIZED  
    SDH::cSDH, 160, 161  
eAS\_POSITIONING  
    SDH::cSDH, 159–161

eAS\_SPEED\_MODE  
SDH::cSDH, 160, 161

eAxisState  
SDH::cSDH, 159, 160

eControllerType  
SDH::cSDHBase, 278, 282

eCT\_DIMENSION  
SDH::cSDHBase, 279, 282

eCT\_INVALID  
SDH::cSDHBase, 279, 282

eCT\_POSE  
SDH::cSDHBase, 279, 282

eCT\_VELOCITY  
SDH::cSDHBase, 279, 282

eCT\_VELOCITY\_ACCELERATION  
SDH::cSDHBase, 279, 282

eEC\_ACCESS\_DENIED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_ALREADY\_OPEN  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_ALREADY\_RUNNING  
SDH::cSDHBase, 277, 280, 281

eEC\_AXIS\_DISABLED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_CHECKSUM\_ERROR  
SDH::cSDHBase, 277, 280, 281

eEC\_CMD\_ABORTED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_CMD\_FAILED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_CMD\_FORMAT\_ERROR  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_CMD\_UNKNOWN  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_DEVICE\_NOT\_FOUND  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_DEVICE\_NOT\_OPENED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_DIMENSION  
SDH::cSDHBase, 277, 278, 281

eEC\_FEATURE\_NOT\_SUPPORTED  
SDH::cSDHBase, 277, 280, 281

eEC\_HOMING\_ERROR  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_INCONSISTENT\_DATA  
SDH::cSDHBase, 277, 280, 281

eEC\_INDEX\_OUT\_OF\_BOUNDS  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_INSUFFICIENT\_RESOURCES  
SDH::cSDHBase, 277, 280, 281

eEC\_INVALID\_HANDLE  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_INVALID\_PARAMETER  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_IO\_ERROR  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_NO\_DATAPIPE  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_NO\_PARAMS\_EXPECTED  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_NO\_SENSOR  
SDH::cSDHBase, 276, 277, 280, 281

eEC\_NOT\_AVAILABLE  
SDH::cSDHBase, 276, 277, 280, 281

eEC\_NOT\_ENOUGH\_PARAMS  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_NOT\_INITIALIZED  
SDH::cSDHBase, 277, 280, 281

eEC\_OVER\_TEMPERATURE  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_RANGE\_ERROR  
SDH::cSDHBase, 277, 278, 280, 281

eEC\_READ\_ERROR  
SDH::cSDHBase, 277, 280, 281

eEC\_SUCCESS  
SDH::cSDHBase, 276, 277, 280, 281

eEC\_TIMEOUT  
SDH::cSDHBase, 277, 280, 281

eEC\_WRITE\_ERROR  
SDH::cSDHBase, 277, 280, 281

eErrorCode  
SDH::cSDHBase, 276, 280

eGID\_CENTRICAL  
SDH::cSDHBase, 278, 281, 282

eGID\_CYLINDRICAL  
SDH::cSDHBase, 278, 282

eGID\_DIMENSION  
SDH::cSDHBase, 278, 282

eGID\_INVALID  
SDH::cSDHBase, 278, 281, 282

eGID\_PARALLEL  
SDH::cSDHBase, 278, 282

eGID\_SPHERICAL  
SDH::cSDHBase, 278, 282

eGraspId  
SDH::cSDHBase, 278, 281

Elapsed  
SDH::cSimpleTime, 350, 351

Elapsed\_us  
SDH::cSimpleTime, 350, 351

eMAX\_CHARS  
SDH::cSimpleStringList, 346

eMAX\_LINES  
SDH::cSimpleStringList, 346

eMAX\_MSG  
SDH::cMsg, 129

eMCM\_DIMENSION  
SDH::cSDH, 159, 160

eMCM\_GRIP  
    SDH::cSDH, 159, 160

eMCM\_HOLD  
    SDH::cSDH, 159, 160

eMCM\_MOVE  
    SDH::cSDH, 159, 160

EmergencyStop  
    SDH::cSDH, 173, 221

eMotorCurrentMode  
    SDH::cSDH, 159, 160

eNUMBER\_OF\_ELEMENTS  
    SDH::cSimpleVector, 353, 354

EOL  
    SDH::cSDHSerial, 331, 332

eps  
    SDH::cSDHBase, 288

eps\_v  
    SDH::cSDHBase, 288

error\_code  
    SDH::cDSA::sControllerInfo, 105  
    SDH::cDSA::sMatrixInfo, 107  
    SDH::cDSA::sSensorInfo, 110

ESD\_strerror  
    cpp.desire.final/sdh/canserial-esd.cpp, 468  
    sdh/canserial-esd.cpp, 470

eVelocityProfile  
    SDH::cSDHBase, 279, 282

eVP\_DIMENSION  
    SDH::cSDHBase, 279, 283

eVP\_INVALID  
    SDH::cSDHBase, 279, 283

eVP\_RAMP  
    SDH::cSDHBase, 279, 283

eVP\_SIN\_SQUARE  
    SDH::cSDHBase, 279, 283

exchange  
    cpp.desire.final/vcc/getopt.c, 604  
    vcc/getopt.c, 607

ExtractFirmwareState  
    SDH::cSDHSerial, 312, 322

f\_max\_acceleration\_v  
    SDH::cSDH, 263, 265

f\_max\_angle\_v  
    SDH::cSDH, 262, 265

f\_max\_motor\_current\_v  
    SDH::cSDH, 262, 265

f\_max\_velocity\_v  
    SDH::cSDH, 262, 265

f\_min\_acceleration\_v  
    SDH::cSDH, 262, 265

f\_min\_angle\_v  
    SDH::cSDH, 262, 265

f\_min\_motor\_current\_v  
    SDH::cSDH, 262, 265

    SDH::cSDH, 262, 265

f\_min\_velocity\_v  
    SDH::cSDH, 262, 265

f\_ones\_v  
    SDH::cSDH, 262, 264

f\_zeros\_v  
    SDH::cSDH, 262, 264

factor  
    SDH::cUnitConverter, 365

fd  
    SDH::cRS232, 140

feature\_flags  
    SDH::cDSA::sControllerInfo, 105  
    SDH::cDSA::sMatrixInfo, 107  
    SDH::cDSA::sSensorInfo, 110

finger\_axis\_index  
    SDH::cSDH, 261, 264

finger\_number\_of\_axes  
    SDH::cSDH, 261, 264

firmware\_error\_codes  
    SDH::cSDHBase, 286, 289

firmware\_state  
    SDH::cSDHBase, 288

first\_nonopt  
    cpp.desire.final/vcc/getopt.c, 604  
    vcc/getopt.c, 607

flag  
    option, 367

flags  
    SDH::cDSA::sTactileSensorFrame, 112

force  
    SDH::cDSA::sContactInfo, 104

force\_factor  
    SDH::cDSA, 102

frame  
    SDH::cDSA, 101

framerate  
    cSDHOOptions, 304

FromString  
    SDH::cSimpleVector, 354, 355

fullframe  
    cSDHOOptions, 304

fullscale  
    SDH::cDSA::sMatrixInfo, 107

g\_sdh\_debug\_log  
    SDH, 52

generated\_by  
    SDH::cDSA::sSensorInfo, 110

get\_duration  
    SDH::cSDHSerial, 312, 322

GetAgeOfFrame  
    SDH::cDSA, 97, 100

GetAxisActualAngle

SDH::cSDH, 185, 186, 233, 234  
 GetAxisActualState  
     SDH::cSDH, 180, 181, 228, 229  
 GetAxisActualVelocity  
     SDH::cSDH, 190, 191, 238, 239  
 GetAxisEnable  
     SDH::cSDH, 179, 180, 227, 228  
 GetAxisLimitAcceleration  
     SDH::cSDH, 189, 190, 237, 238  
 GetAxisLimitVelocity  
     SDH::cSDH, 189, 237  
 GetAxisMaxAcceleration  
     SDH::cSDH, 198, 199, 246, 247  
 GetAxisMaxAngle  
     SDH::cSDH, 196, 197, 244, 245  
 GetAxisMaxVelocity  
     SDH::cSDH, 197, 198, 245, 246  
 GetAxisMinAngle  
     SDH::cSDH, 195, 196, 243, 244  
 GetAxisMotorCurrent  
     SDH::cSDH, 177, 178, 225, 226  
 GetAxisReferenceVelocity  
     SDH::cSDH, 191, 192, 239, 240  
 GetAxisTargetAcceleration  
     SDH::cSDH, 194, 195, 242, 243  
 GetAxisTargetAngle  
     SDH::cSDH, 185, 232, 233  
 GetAxisTargetVelocity  
     SDH::cSDH, 188, 236  
 GetAxisValueVector  
     SDH::cSDH, 165, 212  
 GetContactArea  
     SDH::cDSA, 97, 100  
 GetContactInfo  
     SDH::cDSA, 97, 101  
 GetController  
     SDH::cSDH, 175, 222  
 GetControllerInfo  
     SDH::cDSA, 96, 99  
 GetCRC  
     SDH::cCRC, 78  
 GetCRC\_HB  
     SDH::cCRC, 78  
 GetCRC\_LB  
     SDH::cCRC, 78  
 GetDecimalPlaces  
     SDH::cUnitConverter, 363, 365  
 GetDuration  
     SDH::cSDHSerial, 312, 322  
 GetEps  
     SDH::cSDHBase, 284, 286  
 GetEpsVector  
     SDH::cSDHBase, 284, 286  
 GetFactor  
     SDH::cUnitConverter, 363, 364  
 GetFingerActualAngle  
     SDH::cSDH, 205, 253  
 GetFingerAxisIndex  
     SDH::cSDH, 167, 215  
 GetFingerEnable  
     SDH::cSDH, 202, 203, 250, 251  
 GetFingerMaxAngle  
     SDH::cSDH, 206, 207, 254, 255  
 GetFingerMinAngle  
     SDH::cSDH, 205, 206, 253, 254  
 GetFingerNumberOfAxes  
     SDH::cSDH, 167, 214  
 GetFingerTargetAngle  
     SDH::cSDH, 204, 252  
 GetFingerXYZ  
     SDH::cSDH, 207, 208, 255, 256  
 GetFirmwareRelease  
     SDH::cSDH, 168, 215  
 GetFirmwareState  
     SDH::cSDHBase, 284, 286  
 GetFlag  
     SDH::cDBG, 84, 85  
 GetFrame  
     SDH::cDSA, 96, 99  
 GetGripMaxVelocity  
     SDH::cSDH, 210, 258  
 GetHand  
     cpp.desire.final/demo/demo-mimic.cpp, 412  
     demo/demo-mimic.cpp, 414  
 GetHandle  
     SDH::cCANSerial\_PEAK, 69, 71  
 GetInfo  
     SDH::cSDH, 168, 216  
 GetKind  
     SDH::cUnitConverter, 363, 364  
 GetLastErrorMessage  
     cpp.desire.final/sdh/rs232-vcc.cpp, 532  
     sdh/rs232-vcc.cpp, 534  
 GetLibraryName  
     SDH::cSDH, 168, 215  
 GetLibraryRelease  
     SDH::cSDH, 167, 215  
 GetMatrixIndex  
     SDH::cDSA, 97, 100  
 GetMatrixInfo  
     SDH::cDSA, 96, 99  
 GetMotorCurrentModeFunction  
     SDH::cSDH, 166, 213  
 GetName  
     SDH::cUnitConverter, 363, 364  
 GetNumberOfAxes  
     SDH::cSDHBase, 284, 285  
 GetNumberOfFingers

SDH::cSDHBase, 284, 286  
GetNumberOfTemperatureSensors  
    SDH::cSDHBase, 284, 286  
GetOffset  
    SDH::cUnitConverter, 363, 365  
getopt  
    cpp.desire.final/vcc/getopt.c, 604  
    cpp.desire.final/vcc/getopt.h, 609  
    vcc/getopt.c, 607  
    vcc/getopt.h, 611  
GETOPT\_INTERFACE\_VERSION  
    cpp.desire.final/vcc/getopt.c, 603  
    cpp.desire.final/vcc/getopt1.c, 612  
    vcc/getopt.c, 606  
    vcc/getopt1.c, 613  
getopt\_long  
    cpp.desire.final/vcc/getopt.h, 609  
    cpp.desire.final/vcc/getopt1.c, 612  
    vcc/getopt.h, 611  
    vcc/getopt1.c, 613  
getopt\_long\_only  
    cpp.desire.final/vcc/getopt.h, 609  
    cpp.desire.final/vcc/getopt1.c, 612  
    vcc/getopt.h, 611  
    vcc/getopt1.c, 613  
GetSensorInfo  
    SDH::cDSA, 96, 99  
GetStringFromControllerType  
    SDH::cSDHBase, 284, 285  
GetStringFromErrorCode  
    SDH::cSDHBase, 284, 285  
GetStringFromGraspId  
    SDH::cSDHBase, 284, 285  
GetSymbol  
    SDH::cUnitConverter, 363, 364  
GetTemperature  
    SDH::cSDH, 168, 169, 216, 217  
GetTexel  
    SDH::cDSA, 97, 100  
GetTimeout  
    SDH::cSerialBase, 338, 340  
GetVelocityProfile  
    SDH::cSDH, 175, 223  
GotoStartPose  
    cpp.desire.final/demo/demo-contact-  
        grasping.cpp, 388  
    demo/demo-contact-grasping.cpp, 391  
grasp\_id\_name  
    SDH::cSDHBase, 287, 289  
grip  
    SDH::cSDHSerial, 321, 331  
grip\_max\_velocity  
    SDH::cSDH, 263  
GripHand  
    SDH::cSDH, 210, 258  
h  
    SDH::cSDH, 263  
handle  
    SDH::cCANSerial\_PEAK, 72  
has\_arg  
    option, 367  
hw\_revision  
    SDH::cDSA::sMatrixInfo, 107  
    SDH::cDSA::sSensorInfo, 110  
hw\_version  
    SDH::cDSA::sControllerInfo, 105  
id  
    SDH::cSDHSerial, 319, 329  
id\_read  
    cSDHOptions, 304  
    SDH::cCANSerial\_ESD, 60, 61  
    SDH::cCANSerial\_PEAK, 72  
id\_write  
    cSDHOptions, 304  
    SDH::cCANSerial\_ESD, 61  
    SDH::cCANSerial\_PEAK, 72  
igrip  
    SDH::cSDHSerial, 320, 330  
ihold  
    SDH::cSDHSerial, 320, 330  
ilim  
    SDH::cSDHSerial, 313, 324  
InIndex  
    SDH, 49  
initial\_value  
    SDH::cCRC, 78  
InRange  
    SDH, 49  
Int16  
    SDH, 47  
Int32  
    SDH, 47  
Int8  
    SDH, 47  
interrupt  
    SDH::cDSAUpdater, 119  
io\_set\_old  
    SDH::cRS232, 140  
IsGrasped  
    SDH::cIsGraspedBase, 122  
    SDH::cIsGraspedByArea, 127  
IsOpen  
    SDH::cCANSerial\_ESD, 58, 59  
    SDH::cCANSerial\_PEAK, 70, 71  
    SDH::cRS232, 136–139  
    SDH::cSDH, 173, 220

SDH::cSDHBase, 285, 286  
 SDH::cSDHSerial, 312, 322  
 SDH::cSerialBase, 338, 339  
 IsVirtualAxis  
     SDH::cSDH, 166, 214  
 kind  
     SDH::cUnitConverter, 365  
 kv  
     SDH::cSDHSerial, 313, 323  
 11  
     SDH::cSDH, 263  
 12  
     SDH::cSDH, 263  
 last\_nonopt  
     cpp.desire.final/vcc/getopt.c, 604  
     vcc/getopt.c, 607  
 Length  
     SDH::cSimpleStringList, 347  
 line  
     SDH::cSimpleStringList, 348  
 m  
     SDH::cSDHSerial, 316, 326  
 M\_CMSG\_MSG  
     cpp.desire.final/sdh/canserial-peak.h, 480  
     sdh/canserial-peak.h, 482  
 m\_device  
     SDH::cCANSerial\_PEAK, 72  
 M\_PI  
     SDH, 52  
 m\_sequetime  
     SDH::cSDHSerial, 331  
 main  
     cpp.desire.final/demo/cancat.cpp, 384  
     cpp.desire.final/demo/demo-contact-grasping.cpp, 388  
     cpp.desire.final/demo/demo-dsa.cpp, 394  
     cpp.desire.final/demo/demo-GetAxisActualAngle.cpp, 400  
     cpp.desire.final/demo/demo-GetFingerXYZ.cpp, 406  
     cpp.desire.final/demo/demo-mimic.cpp, 412  
     cpp.desire.final/demo/demo-simple-withtiming.cpp, 416  
     cpp.desire.final/demo/demo-simple.cpp, 420  
     cpp.desire.final/demo/demo-simple2.cpp, 424  
     cpp.desire.final/demo/demo-simple3.cpp, 428  
     cpp.desire.final/demo/demo-temperature.cpp, 432  
     cpp.desire.final/demo/demo-velocity-acceleration.cpp, 436  
     demo-griphand.cpp, 615  
 demo/cancat.cpp, 386  
 demo/demo-contact-grasping.cpp, 391  
 demo/demo-dsa.cpp, 397  
 demo/demo-GetAxisActualAngle.cpp, 403  
 demo/demo-GetFingerXYZ.cpp, 409  
 demo/demo-mimic.cpp, 414  
 demo/demo-simple-withtiming.cpp, 418  
 demo/demo-simple.cpp, 422  
 demo/demo-simple2.cpp, 426  
 demo/demo-simple3.cpp, 430  
 demo/demo-temperature.cpp, 434  
 demo/demo-velocity-acceleration.cpp, 438  
 Makefile, 633  
 map  
     SDH, 49  
 matrix\_center\_x  
     SDH::cDSA::sMatrixInfo, 107  
 matrix\_center\_y  
     SDH::cDSA::sMatrixInfo, 107  
 matrix\_center\_z  
     SDH::cDSA::sMatrixInfo, 107  
 matrix\_info  
     SDH::cDSA, 101, 102  
 matrix\_theta\_x  
     SDH::cDSA::sMatrixInfo, 107  
 matrix\_theta\_y  
     SDH::cDSA::sMatrixInfo, 107  
 matrix\_theta\_z  
     SDH::cDSA::sMatrixInfo, 107  
 matrixinfo  
     cSDHOPTIONS, 304  
 max\_angle\_v  
     SDH::cSDHBase, 288  
 MAX\_DEV\_LENGTH  
     cSDHOPTIONS, 304  
 max\_payload\_size  
     SDH::cDSA::sResponse, 109  
 min\_angle\_v  
     SDH::cSDHBase, 288  
 MoveAxis  
     SDH::cSDH, 199, 200, 247, 248  
 MoveFinger  
     SDH::cSDH, 208, 210, 256, 258  
 MoveHand  
     SDH::cSDH, 210, 258  
 msg  
     SDH::cMsg, 130  
     SDH::cSDHLibraryException, 299  
 my\_index  
     cpp.desire.final/vcc/getopt.c, 604  
     vcc/getopt.c, 607  
 name  
     option, 367

SDH::cUnitConverter, 365  
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, 261  
nb\_cells  
SDH::cDSA, 101  
nb\_lines\_to\_ignore  
SDH::cSDHSerial, 332  
nb\_matrices  
SDH::cDSA::sSensorInfo, 110  
net  
cSDHOptions, 304  
SDH::cCANSerial\_ESD, 60  
nextchar  
cpp.desire.final/vcc/getopt.c, 604  
vcc/getopt.c, 607  
NextLine  
SDH::cSimpleStringList, 347  
no\_argument  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.h, 611  
NOMINMAX  
cpp.desire.final/sdh/canserial-esd.h, 472  
sdh/canserial-esd.h, 474  
NONOPTION\_P  
cpp.desire.final/vcc/getopt.c, 603  
vcc/getopt.c, 606  
normal\_color  
SDH::cDBG, 86  
NTCAN\_HANDLE  
SDH, 47  
ntcan\_handle  
SDH::cCANSerial\_ESD, 61  
NULL  
cpp.desire.final/vcc/getopt1.c, 612  
vcc/getopt1.c, 613  
numaxis  
SDH::cSDHSerial, 320, 330  
NUMBER\_OF\_AXES  
SDH::cSDHBase, 288  
NUMBER\_OF\_AXES\_PER\_FINGER  
SDH::cSDH, 261  
NUMBER\_OF\_FINGERS  
SDH::cSDHBase, 288  
NUMBER\_OF\_TEMPERATURE\_SENSORS  
SDH::cSDHBase, 288  
NUMBER\_OF\_VIRTUAL\_AXES  
SDH::cSDH, 261  
NumerifyRelease  
SDH, 50  
offset  
SDH::cSDH, 263, 265  
SDH::cUnitConverter, 365  
ones\_v  
SDH::cSDH, 262, 265  
Online help of demonstration programs, 27  
Open  
SDH::cCANSerial\_ESD, 58, 59  
SDH::cCANSerial\_PEAK, 69, 71  
SDH::cDSA, 96, 99  
SDH::cRS232, 136–139  
SDH::cSDHSerial, 311, 321  
SDH::cSerialBase, 337, 339  
OpenCAN\_ESD  
SDH::cSDH, 170, 171, 218  
OpenCAN\_PEAK  
SDH::cSDH, 171, 172, 219  
OpenRS232  
SDH::cSDH, 169, 217  
operator<<  
SDH, 50, 51  
SDH::cDSA, 101  
optarg  
cpp.desire.final/vcc/getopt.c, 604  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.c, 607  
vcc/getopt.h, 611  
opterr  
cpp.desire.final/vcc/getopt.c, 604  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.c, 607  
vcc/getopt.h, 611  
optind  
cpp.desire.final/vcc/getopt.c, 604  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.c, 607  
vcc/getopt.h, 611  
option, 367  
flag, 367  
has\_arg, 367  
name, 367  
val, 367  
optional\_argument  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.h, 611  
 getopt  
cpp.desire.final/vcc/getopt.c, 604  
cpp.desire.final/vcc/getopt.h, 609  
vcc/getopt.c, 607  
vcc/getopt.h, 611  
ordering  
cpp.desire.final/vcc/getopt.c, 604

vcc/getopt.c, 607  
 output  
     SDH::cDBG, 86  
  
 p  
     SDH::cSDHSerial, 316, 326  
 packet\_id  
     SDH::cDSA::sResponse, 109  
 Parse  
     cSDHOptions, 301, 302  
 ParseFrame  
     SDH::cDSA, 95, 99  
 payload  
     SDH::cDSA::sResponse, 109  
 PCAN\_HANDLE  
     SDH, 47  
 PDM  
     SDH::cDBG, 85  
 pDoubleUnitConverterFunction  
     SDH, 47  
 PEAK\_sterror  
     cpp.desire.final/sdh/canserial-peak.cpp, 476  
     sdh/canserial-peak.cpp, 478  
 period  
     cSDHOptions, 304  
 PERMUTE  
     cpp.desire.final/vcc/getopt.c, 603  
     vcc/getopt.c, 606  
 pGetFunction  
     SDH, 47  
 pid  
     SDH::cSDHSerial, 313, 323  
 port  
     SDH::cRS232, 140  
 pos  
     SDH::cSDHSerial, 317, 327  
 pos\_save  
     SDH::cSDHSerial, 317, 327  
 posixly\_correct  
     cpp.desire.final/vcc/getopt.c, 604  
     vcc/getopt.c, 607  
 power  
     SDH::cSDHSerial, 314, 324  
 PRINT\_MEMBER  
     cpp.desire.final/sdh/dsa.cpp, 494  
     sdh/dsa.cpp, 496  
 PRINT\_MEMBER\_HEX  
     cpp.desire.final/sdh/dsa.cpp, 494  
     sdh/dsa.cpp, 496  
 PROJECT\_COPYRIGHT  
     cpp.desire.final/sdh/release.h, 504  
     sdh/release.h, 513  
 PROJECT\_DATE  
     cpp.desire.final/sdh/release.h, 504  
     sdh/release.h, 513  
 PROJECT\_NAME  
     cpp.desire.final/sdh/release.h, 504  
     sdh/release.h, 513  
 PROJECT\_RELEASE  
     cpp.desire.final/sdh/release.h, 504  
     sdh/release.h, 514  
 property  
     SDH::cSDHSerial, 314, 324  
 pSetFunction  
     SDH, 47  
  
 QueryControllerInfo  
     SDH::cDSA, 95, 98  
 QueryMatrixInfo  
     SDH::cDSA, 95, 98  
 QueryMatrixInfos  
     SDH::cDSA, 95, 99  
 QuerySensorInfo  
     SDH::cDSA, 95, 98  
  
 RadToDeg  
     SDH, 51  
 Read  
     SDH::cCANSerial\_ESD, 59, 60  
     SDH::cCANSerial\_PEAk, 70, 71  
     SDH::cRS232, 136–139  
     SDH::cSerialBase, 338, 340  
 read\_timeout\_us  
     SDH::cDSA, 102  
 ReadControllerInfo  
     SDH::cDSA, 95, 98  
 ReadFrame  
     SDH::cDSA, 95, 98  
 readline  
     SDH::cRS232, 138, 140  
     SDH::cSerialBase, 339, 340  
 ReadMatrixInfo  
     SDH::cDSA, 95, 98  
 ReadResponse  
     SDH::cDSA, 94, 98  
 ReadSensorInfo  
     SDH::cDSA, 95, 98  
 ref  
     SDH::cSDHSerial, 317, 327  
 reply  
     SDH::cSDHSerial, 332  
 REQUIRE\_ORDER  
     cpp.desire.final/vcc/getopt.c, 603  
     vcc/getopt.c, 606  
 required\_argument  
     cpp.desire.final/vcc/getopt.h, 609  
     vcc/getopt.h, 611  
 reserved

SDH::cDSA::sMatrixInfo, 107  
Reset  
  SDH::cCRC, 78  
  SDH::cSimpleStringList, 347, 348  
RETURN\_IN\_ORDER  
  cpp.desire.final/vcc/getopt.c, 603  
  vcc/getopt.c, 606  
rs232\_baudrate  
  cSDHOptions, 304  
rvel  
  SDH::cSDHSerial, 318, 328  
  
SDH, 43  
  apply, 48  
  Approx, 48  
  CompareReleases, 48  
  DegToRad, 49  
  g\_sdh\_debug\_log, 52  
  InIndex, 49  
  InRange, 49  
  Int16, 47  
  Int32, 47  
  Int8, 47  
  M\_PI, 52  
  map, 49  
  NTCAN\_HANDLE, 47  
  NumerifyRelease, 50  
  operator<<, 50, 51  
  PCAN\_HANDLE, 47  
  pDoubleUnitConverterFunction, 47  
  pGetFunction, 47  
  pSetFunction, 47  
  RadToDeg, 51  
  SleepSec, 51  
  tCRCValue, 47  
  ToRange, 51  
  uc\_identity, 52  
  UInt16, 47  
  UInt32, 47  
  UInt8, 48  
sdh/basisdef.h, 465  
  SDH\_ASSERT\_TYPESIZES, 466  
sdh/canserial-esd.cpp, 469  
  ESD\_strerror, 470  
sdh/canserial-esd.h, 473  
  CAN\_ESD\_RXQUEUESIZE, 474  
  CAN\_ESD\_TXQUEUESIZE, 474  
  NOMINMAX, 474  
sdh/canserial-peak.cpp, 477  
  PEAK\_strerror, 478  
  USE\_HANDLE, 478  
  USE\_HANDLES, 478  
sdh/canserial-peak.h, 481  
  M\_CMSG\_MSG, 482  
sdh/crc.cpp, 484  
sdh/crc.h, 487  
sdh/dbg.h, 491  
  V, 492  
  VAR, 492  
sdh/dsa.cpp, 495  
  PRINT\_MEMBER, 496  
  PRINT\_MEMBER\_HEX, 496  
  SDH\_NAMESPACE\_PREFIX, 496  
sdh/dsa.h, 500  
  DSA\_MAX\_PREAMBLE\_SEARCH, 502  
sdh/release.h, 513  
  PROJECT\_COPYRIGHT, 513  
  PROJECT\_DATE, 513  
  PROJECT\_NAME, 513  
  PROJECT\_RELEASE, 514  
sdh/rs232-cygwin.cpp, 525  
  DBG, 526  
  SDH\_RS232\_CYGWIN\_DEBUG, 526  
  SDH\_RS232\_CYGWIN\_DEBUG\_ASCII, 526  
  StrDupNew, 526  
sdh/rs232-cygwin.h, 529  
sdh/rs232-vcc.cpp, 533  
  \_CRT\_SECURE\_NO\_WARNINGS, 534  
  DBG, 534  
  GetLastErrorMessage, 534  
  SDH\_RS232\_VCC\_DEBUG, 534  
  SDH\_RS232\_VCC\_DEBUG\_ASCII, 534  
sdh/rs232-vcc.h, 537  
  SDH\_RS232\_VCC\_ASYNC, 538  
sdh/sdh.cpp, 540  
  \_USE\_MATH\_DEFINES, 541  
sdh/sdh.h, 544  
sdh/sdhbase.cpp, 547  
sdh/sdhbase.h, 550  
sdh/sdhexception.cpp, 553  
sdh/sdhexception.h, 556  
sdh/sdhllibrary\_settings.h, 559  
sdh/sdhserial.cpp, 561  
sdh/sdhserial.h, 564  
sdh/serialbase.cpp, 567  
sdh/serialbase.h, 570  
sdh/simplestringlist.cpp, 573  
sdh/simplestringlist.h, 576  
sdh/simpletime.h, 579  
sdh/simplevector.cpp, 581  
sdh/simplevector.h, 584  
sdh/unit\_converter.cpp, 586  
sdh/unit\_converter.h, 589  
sdh/util.cpp, 592  
  \_USE\_MATH\_DEFINES, 593  
sdh/util.h, 597  
  DEFINE\_TO\_CASECOMMAND, 598

DEFINE\_TO\_CASECOMMAND\_MSG, 599  
 SDH::cCANSerial\_ESD, 53  
     baudrate, 60  
     BaudrateToBaudrateCode, 58, 59  
     cCANSerial\_ESD, 57, 58  
     Close, 58, 59  
     id\_read, 60, 61  
     id\_write, 61  
     IsOpen, 58, 59  
     net, 60  
     ntcan\_handle, 61  
     Open, 58, 59  
     Read, 59, 60  
     SetTimeout, 59, 60  
     status, 61  
     write, 58, 60  
 SDH::cCANSerial\_ESDEception, 62  
     cCANSerial\_ESDEception, 64  
 SDH::cCANSerial\_PEAK, 65  
     baudrate, 72  
     BaudrateToBaudrateCode, 69, 70  
     cCANSerial\_PEAK, 68, 69  
     Close, 70, 71  
     GetHandle, 69, 71  
     handle, 72  
     id\_read, 72  
     id\_write, 72  
     IsOpen, 70, 71  
     m\_device, 72  
     Open, 69, 71  
     Read, 70, 71  
     SetTimeout, 70, 72  
     write, 70, 71  
 SDH::cCANSerial\_PEAKException, 73  
     cCANSerial\_PEAKException, 75  
 SDH::cCRC, 76  
     AddByte, 78  
     cCRC, 77  
     crc\_table, 79  
     current\_crc, 78  
     GetCRC, 78  
     GetCRC\_HB, 78  
     GetCRC\_LB, 78  
     initial\_value, 78  
     Reset, 78  
 SDH::cCRC\_DSACON32m, 80  
     cCRC\_DSACON32m, 81  
     crc\_table\_dsacon32m, 81  
 SDH::cDBG, 83  
     ~cDBG, 84  
     cDBG, 84  
     debug\_color, 86  
     debug\_flag, 86  
     GetFlag, 84, 85  
     normal\_color, 86  
     output, 86  
     PDM, 85  
     SetColor, 85  
     SetFlag, 84, 85  
     SetOutput, 85  
 SDH::cDSA, 87  
     ~cDSA, 94  
     \_\_packed\_\_, 101  
     calib\_pressure, 102  
     calib\_voltage, 102  
     cDSA, 93, 94  
     Close, 96, 99  
     comm\_interface, 101  
     contact\_area\_cell\_threshold, 102  
     contact\_force\_cell\_threshold, 102  
     controller\_info, 101  
     dbg, 101  
     do\_RLE, 101  
     force\_factor, 102  
     frame, 101  
     GetAgeOfFrame, 97, 100  
     GetContactArea, 97, 100  
     GetContactInfo, 97, 101  
     GetControllerInfo, 96, 99  
     GetFrame, 96, 99  
     GetMatrixIndex, 97, 100  
     GetMatrixInfo, 96, 99  
     GetSensorInfo, 96, 99  
     GetTexel, 97, 100  
     matrix\_info, 101, 102  
     nb\_cells, 101  
     Open, 96, 99  
     operator<<, 101  
     ParseFrame, 95, 99  
     QueryControllerInfo, 95, 98  
     QueryMatrixInfo, 95, 98  
     QueryMatrixInfos, 95, 99  
     QuerySensorInfo, 95, 98  
     read\_timeout\_us, 102  
     ReadControllerInfo, 95, 98  
     ReadFrame, 95, 98  
     ReadMatrixInfo, 95, 98  
     ReadResponse, 94, 98  
     ReadSensorInfo, 95, 98  
     sensor\_info, 101  
     SetFramerate, 96, 99  
     SetFramerateRetries, 96, 100  
     start\_dsa, 102  
     start\_pc, 102  
     texel\_offset, 102  
     tTexel, 93  
     UpdateFrame, 96, 99  
     WriteCommand, 94, 98

WriteCommandWithPayload, 94, 97  
SDH::cDSA::sContactInfo, 104  
    area, 104  
    cog\_x, 104  
    cog\_y, 104  
    force, 104  
SDH::cDSA::sControllerInfo, 105  
    active\_interface, 105  
    can\_baudrate, 105  
    can\_id, 105  
    error\_code, 105  
    feature\_flags, 105  
    hw\_version, 105  
    senscon\_type, 105  
    serial\_no, 105  
    status\_flags, 105  
    sw\_version, 105  
SDH::cDSA::sMatrixInfo, 106  
    cells\_x, 107  
    cells\_y, 107  
    error\_code, 107  
    feature\_flags, 107  
    fullscale, 107  
    hw\_revision, 107  
    matrix\_center\_x, 107  
    matrix\_center\_y, 107  
    matrix\_center\_z, 107  
    matrix\_theta\_x, 107  
    matrix\_theta\_y, 107  
    matrix\_theta\_z, 107  
    reserved, 107  
    texel\_height, 107  
    texel\_width, 107  
    uid, 107  
SDH::cDSA::sResponse, 108  
    max\_payload\_size, 109  
    packet\_id, 109  
    payload, 109  
    size, 109  
    sResponse, 108  
SDH::cDSA::sSensorInfo, 110  
    error\_code, 110  
    feature\_flags, 110  
    generated\_by, 110  
    hw\_revision, 110  
    nb\_matrices, 110  
    serial\_no, 110  
SDH::cDSA::sTactileSensorFrame, 111  
    flags, 112  
    sTactileSensorFrame, 111  
    texel, 112  
    timestamp, 112  
SDH::cDSAException, 113  
    cDSAException, 115  
                SDH::cDSAUpdater, 116  
                cDSAUpdater, 118  
                DEFAULT\_ERROR\_THRESHOLD, 119  
                interrupt, 119  
                SDH::cIsGraspedBase, 120  
                cIsGraspedBase, 122  
                IsGrasped, 122  
                ts, 122  
                SDH::cIsGraspedByArea, 124  
                cIsGraspedByArea, 126  
                IsGrasped, 127  
                SetCondition, 126, 127  
                SDH::cMsg, 128  
                c\_str, 130  
                cMsg, 129  
                eMAX\_MSG, 129  
                msg, 130  
                SDH::cRS232, 131  
                ~cRS232, 135  
                baudrate, 140  
                BaudrateToBaudrateCode, 136, 138  
                Close, 136–139  
                cRS232, 134, 135  
                device\_format\_string, 140  
                fd, 140  
                io\_set\_old, 140  
                IsOpen, 136–139  
                Open, 136–139  
                port, 140  
                Read, 136–139  
                readline, 138, 140  
                SetTimeout, 137, 139  
                status, 140  
                write, 136–139  
                SDH::cRS232Exception, 141  
                cRS232Exception, 143  
                SDH::cSDH, 144  
                ~cSDH, 163, 164  
                \_GetFingerXYZ, 166, 213  
                all\_axes, 263, 266  
                all\_fingers, 263, 266  
                all\_real\_axes, 263, 266  
                all\_temperature\_sensors, 263, 266  
                Close, 172, 220  
                com, 263, 266  
                comm\_interface, 263  
                cSDH, 161, 163  
                d, 263  
                eAS\_CCW\_BLOCKED, 160, 161  
                eAS\_CW\_BLOCKED, 160, 161  
                eAS\_DIMENSION, 160, 161  
                eAS\_DISABLED, 160, 161  
                eAS\_IDLE, 159–161  
                eAS\_LIMITS\_REACHED, 160, 161

eAS\_NOT\_INITIALIZED, 160, 161  
eAS\_POSITIONING, 159–161  
eAS\_SPEED\_MODE, 160, 161  
eAxisState, 159, 160  
eMCM\_DIMENSION, 159, 160  
eMCM\_GRIP, 159, 160  
eMCM\_HOLD, 159, 160  
eMCM\_MOVE, 159, 160  
EmergencyStop, 173, 221  
eMotorCurrentMode, 159, 160  
f\_max\_acceleration\_v, 263, 265  
f\_max\_angle\_v, 262, 265  
f\_max\_motor\_current\_v, 262, 265  
f\_max\_velocity\_v, 262, 265  
f\_min\_acceleration\_v, 262, 265  
f\_min\_angle\_v, 262, 265  
f\_min\_motor\_current\_v, 262, 265  
f\_min\_velocity\_v, 262, 265  
f\_ones\_v, 262, 264  
f\_zeros\_v, 262, 264  
finger\_axis\_index, 261, 264  
finger\_number\_of\_axes, 261, 264  
GetAxisActualAngle, 185, 186, 233, 234  
GetAxisActualState, 180, 181, 228, 229  
GetAxisActualVelocity, 190, 191, 238, 239  
GetAxisEnable, 179, 180, 227, 228  
GetAxisLimitAcceleration, 189, 190, 237, 238  
GetAxisLimitVelocity, 189, 237  
GetAxisMaxAcceleration, 198, 199, 246, 247  
GetAxisMaxAngle, 196, 197, 244, 245  
GetAxisMaxVelocity, 197, 198, 245, 246  
GetAxisMinAngle, 195, 196, 243, 244  
GetAxisMotorCurrent, 177, 178, 225, 226  
GetAxisReferenceVelocity, 191, 192, 239, 240  
GetAxisTargetAcceleration, 194, 195, 242, 243  
GetAxisTargetAngle, 185, 232, 233  
GetAxisTargetVelocity, 188, 236  
GetAxisValueVector, 165, 212  
GetController, 175, 222  
GetFingerActualAngle, 205, 253  
GetFingerAxisIndex, 167, 215  
GetFingerEnable, 202, 203, 250, 251  
GetFingerMaxAngle, 206, 207, 254, 255  
GetFingerMinAngle, 205, 206, 253, 254  
GetFingerNumberOfAxes, 167, 214  
GetFingerTargetAngle, 204, 252  
GetFingerXYZ, 207, 208, 255, 256  
GetFirmwareRelease, 168, 215  
GetGripMaxVelocity, 210, 258  
GetInfo, 168, 216  
GetLibraryName, 168, 215  
GetLibraryRelease, 167, 215  
GetMotorCurrentModeFunction, 166, 213  
GetTemperature, 168, 169, 216, 217  
GetVelocityProfile, 175, 223  
grip\_max\_velocity, 263  
GripHand, 210, 258  
h, 263  
IsOpen, 173, 220  
IsVirtualAxis, 166, 214  
l1, 263  
l2, 263  
MoveAxis, 199, 200, 247, 248  
MoveFinger, 208, 210, 256, 258  
MoveHand, 210, 258  
nb\_all\_axes, 261  
NUMBER\_OF\_AXES\_PER\_FINGER, 261  
NUMBER\_OF\_VIRTUAL\_AXES, 261  
offset, 263, 265  
ones\_v, 262, 265  
OpenCAN\_ESD, 170, 171, 218  
OpenCAN\_PEAK, 171, 172, 219  
OpenRS232, 169, 217  
SetAxisEnable, 178, 179, 226, 227  
SetAxisMotorCurrent, 176, 177, 223, 225  
SetAxisTargetAcceleration, 192, 194, 240, 242  
SetAxisTargetAngle, 183, 184, 231, 232  
SetAxisTargetVelocity, 186, 188, 234, 236  
SetAxisValueVector, 164, 212  
SetController, 174, 221  
SetDebugOutput, 164, 212  
SetFingerEnable, 201, 202, 249, 250  
SetFingerTargetAngle, 203, 204, 251, 252  
SetVelocityProfile, 175, 223  
Stop, 173, 221  
ToIndexVector, 165, 213  
uc\_angle, 264, 266  
uc\_angle\_degrees, 260  
uc\_angle\_radians, 260  
uc\_angular\_acceleration, 264, 266  
uc\_angular\_acceleration\_degrees\_per\_second\_squared, 261  
uc\_angular\_acceleration\_radians\_per\_second\_squared, 261  
uc\_angular\_velocity, 264, 266  
uc\_angular\_velocity\_degrees\_per\_second, 260  
uc\_angular\_velocity\_radians\_per\_second, 260  
uc\_motor\_current, 264, 266  
uc\_motor\_current\_ampere, 261  
uc\_motor\_current\_millampere, 261  
uc\_position, 264, 266  
uc\_position\_meter, 261  
uc\_position\_millimeter, 261  
uc\_temperature, 264, 266  
uc\_temperature\_celsius, 260  
uc\_temperature\_fahrenheit, 260  
uc\_time, 264, 266

uc\_time\_milliseconds, 260  
uc\_time\_seconds, 260  
UseDegrees, 166, 214  
UseRadians, 166, 214  
WaitAxis, 181, 183, 229, 231  
zeros\_v, 262, 265  
SDH::cSDHBase, 268  
  ~cSDHBase, 283  
  All, 276, 280  
  all\_axes\_used, 288  
  cdbg, 286  
  CheckIndex, 283, 285  
  CheckRange, 283–285  
  controller\_type\_name, 287, 289  
  cSDHBase, 283  
  debug\_level, 286  
  eControllerType, 278, 282  
  eCT\_DIMENSION, 279, 282  
  eCT\_INVALID, 279, 282  
  eCT\_POSE, 279, 282  
  eCT\_VELOCITY, 279, 282  
  eCT\_VELOCITY\_ACCELERATION, 279, 282  
  eEC\_ACCESS\_DENIED, 277, 278, 280, 281  
  eEC\_ALREADY\_OPEN, 277, 278, 280, 281  
  eEC\_ALREADY\_RUNNING, 277, 280, 281  
  eEC\_AXIS\_DISABLED, 277, 278, 280, 281  
  eEC\_CHECKSUM\_ERROR, 277, 280, 281  
  eEC\_CMD\_ABORTED, 277, 278, 280, 281  
  eEC\_CMD\_FAILED, 277, 278, 280, 281  
  eEC\_CMD\_FORMAT\_ERROR, 277, 278, 280, 281  
  eEC\_CMD\_UNKNOWN, 277, 278, 280, 281  
  eEC\_DEVICE\_NOT\_FOUND, 277, 278, 280, 281  
  eEC\_DEVICE\_NOT\_OPENED, 277, 278, 280, 281  
  eEC\_DIMENSION, 277, 278, 281  
  eEC\_FEATURE\_NOT\_SUPPORTED, 277, 280, 281  
  eEC\_HOMING\_ERROR, 277, 278, 280, 281  
  eEC\_INCONSISTENT\_DATA, 277, 280, 281  
  eEC\_INDEX\_OUT\_OF\_BOUNDS, 277, 278, 280, 281  
  eEC\_INSUFFICIENT\_RESOURCES, 277, 280, 281  
  eEC\_INVALID\_HANDLE, 277, 278, 280, 281  
  eEC\_INVALID\_PARAMETER, 277, 278, 280, 281  
  eEC\_IO\_ERROR, 277, 278, 280, 281  
  eEC\_NO\_DATAPIPE, 277, 278, 280, 281  
  eEC\_NO\_PARAMS\_EXPECTED, 277, 278, 280, 281  
  eEC\_NO\_SENSOR, 276, 277, 280, 281  
eEC\_NOT\_AVAILABLE, 276, 277, 280, 281  
eEC\_NOT\_ENOUGH\_PARAMS, 277, 278, 280, 281  
eEC\_NOT\_INITIALIZED, 277, 280, 281  
eEC\_OVER\_TEMPERATURE, 277, 278, 280, 281  
eEC\_RANGE\_ERROR, 277, 278, 280, 281  
eEC\_READ\_ERROR, 277, 280, 281  
eEC\_SUCCESS, 276, 277, 280, 281  
eEC\_TIMEOUT, 277, 280, 281  
eEC\_WRITE\_ERROR, 277, 280, 281  
eErrorCode, 276, 280  
eGID\_CENTRICAL, 278, 281, 282  
eGID\_CYLINDRICAL, 278, 282  
eGID\_DIMENSION, 278, 282  
eGID\_INVALID, 278, 281, 282  
eGID\_PARALLEL, 278, 282  
eGID\_SPHERICAL, 278, 282  
eGraspId, 278, 281  
eps, 288  
eps\_v, 288  
eVelocityProfile, 279, 282  
eVP\_DIMENSION, 279, 283  
eVP\_INVALID, 279, 283  
eVP\_RAMP, 279, 283  
eVP\_SIN\_SQUARE, 279, 283  
firmware\_error\_codes, 286, 289  
firmware\_state, 288  
GetEps, 284, 286  
GetEpsVector, 284, 286  
GetFirmwareState, 284, 286  
GetNumberOfAxes, 284, 285  
GetNumberOfFingers, 284, 286  
GetNumberOfTemperatureSensors, 284, 286  
GetStringFromControllerType, 284, 285  
GetStringFromErrorCode, 284, 285  
GetStringFromGraspId, 284, 285  
grasp\_id\_name, 287, 289  
IsOpen, 285, 286  
max\_angle\_v, 288  
min\_angle\_v, 288  
NUMBER\_OF\_AXES, 288  
NUMBER\_OF\_FINGERS, 288  
NUMBER\_OF\_TEMPERATURE\_SENSORS, 288  
SetDebugOutput, 285, 286  
SDH::cSDHErrorCommunication, 290  
  cSDHErrorCommunication, 292  
SDH::cSDHErrorInvalidParameter, 293  
  cSDHErrorInvalidParameter, 295  
SDH::cSDHLibraryException, 296  
  cSDHLibraryException, 297, 298  
msg, 299  
what, 299

SDH::cSDHSerial, 306  
   ~cSDHSerial, 311  
   a, 316, 326  
   alim, 315, 325  
   AxisCommand, 313, 323  
   Close, 312, 322  
   com, 331, 332  
   con, 317, 327  
   cSDHSerial, 311  
   debug, 315, 325  
   demo, 314, 324  
   EOL, 331, 332  
   ExtractFirmwareState, 312, 322  
   get\_duration, 312, 322  
   GetDuration, 312, 322  
   grip, 321, 331  
   id, 319, 329  
   igrip, 320, 330  
   ihold, 320, 330  
   ilim, 313, 324  
   IsOpen, 312, 322  
   kv, 313, 323  
   m, 316, 326  
   m\_sequetime, 331  
   nb\_lines\_to\_ignore, 332  
   numaxis, 320, 330  
   Open, 311, 321  
   p, 316, 326  
   pid, 313, 323  
   pos, 317, 327  
   pos\_save, 317, 327  
   power, 314, 324  
   property, 314, 324  
   ref, 317, 327  
   reply, 332  
   rvel, 318, 328  
   selgrip, 321, 331  
   Send, 312, 322  
   sn, 319, 329  
   soc, 320, 330  
   soc\_date, 320, 330  
   state, 318, 328  
   stop, 316, 326  
   Sync, 312, 322  
   SyncUnknown, 313, 322  
   temp, 319, 329  
   temp\_electronics, 319, 329  
   terminal, 315, 325  
   user\_errors, 314, 324  
   v, 315, 325  
   vel, 318, 328  
   ver, 319, 329  
   ver\_date, 319, 329  
   vlim, 315, 325  
   vp, 316, 326  
 SDH::cSerialBase, 333  
   ~cSerialBase, 337  
   Close, 338, 339  
   cSerialBase, 337  
   dbg, 341  
   GetTimeout, 338, 340  
   IsOpen, 338, 339  
   Open, 337, 339  
   Read, 338, 340  
   readline, 339, 340  
   SetTimeout, 338, 339  
   timeout, 341  
   ungetch, 341  
   ungetch\_valid, 341  
   write, 338, 340  
 SDH::cSerialBaseException, 342  
   cSerialBaseException, 344  
 SDH::cSimpleStringList, 345  
   cSimpleStringList, 347  
   current\_line, 348  
   CurrentLine, 347  
   eMAX\_CHARS, 346  
   eMAX\_LINES, 346  
   Length, 347  
   line, 348  
   NextLine, 347  
   Reset, 347, 348  
 SDH::cSimpleTime, 349  
   a\_time, 351  
   cSimpleTime, 350  
   Elapsed, 350, 351  
   Elapsed\_us, 350, 351  
   StoreNow, 350  
   Timeval, 350, 351  
 SDH::cSimpleVector, 352  
   cSimpleVector, 354  
   eNUMBER\_OF\_ELEMENTS, 353, 354  
   FromString, 354, 355  
   Valid, 355  
   valid, 356  
   value, 356  
   x, 354, 355  
   y, 355  
   z, 355  
 SDH::cSimpleVectorException, 357  
   cSimpleVectorException, 359  
 SDH::cUnitConverter, 360  
   cUnitConverter, 362  
   decimal\_places, 365  
   factor, 365  
   GetDecimalPlaces, 363, 365  
   GetFactor, 363, 364  
   GetKind, 363, 364

GetName, 363, 364  
GetOffset, 363, 365  
GetSymbol, 363, 364  
kind, 365  
name, 365  
offset, 365  
symbol, 365  
ToExternal, 362–364  
ToInternal, 363, 364  
**SDH\_ASSERT\_TYPESIZES**  
  cpp.desire.final/sdh/basisdef.h, 464  
  sdh/basisdef.h, 466  
**sdh\_canpeak\_device**  
  cSDHOptions, 304  
**SDH\_NAMESPACE\_PREFIX**  
  cpp.desire.final/sdh/dsa.cpp, 494  
  sdh/dsa.cpp, 496  
**SDH\_RS232\_CYGWIN\_DEBUG**  
  cpp.desire.final/sdh/rs232-cygwin.cpp, 524  
  sdh/rs232-cygwin.cpp, 526  
**SDH\_RS232\_CYGWIN\_DEBUG\_ASCII**  
  cpp.desire.final/sdh/rs232-cygwin.cpp, 524  
  sdh/rs232-cygwin.cpp, 526  
**SDH\_RS232\_VCC\_ASYNC**  
  cpp.desire.final/sdh/rs232-vcc.h, 536  
  sdh/rs232-vcc.h, 538  
**SDH\_RS232\_VCC\_DEBUG**  
  cpp.desire.final/sdh/rs232-vcc.cpp, 532  
  sdh/rs232-vcc.cpp, 534  
**SDH\_RS232\_VCC\_DEBUG\_ASCII**  
  cpp.desire.final/sdh/rs232-vcc.cpp, 532  
  sdh/rs232-vcc.cpp, 534  
**sdh\_rs\_device**  
  cSDHOptions, 304  
**SDH\_USE\_NAMESPACE**  
  sdhlibrary\_cpp\_sdhlibrary\_settings\_h\_-  
    settings\_group, 23  
  sdhlibrary\_cpp.dox, 601  
  sdhlibrary\_cpp\_sdhlibrary\_settings\_h\_derived\_-  
    settings\_group  
    NAMESPACE\_SDH\_END, 24  
    NAMESPACE\_SDH\_START, 24  
    USING\_NAMESPACE\_SDH, 24  
  sdhlibrary\_cpp\_sdhlibrary\_settings\_h\_settings\_-  
    group  
    SDH\_USE\_NAMESPACE, 23  
  sdhoptions\_long\_options  
    cpp.desire.final/demo/sdhoptions.cpp, 448  
    demo/sdhoptions.cpp, 454  
  sdhoptions\_short\_options  
    cpp.desire.final/demo/sdhoptions.cpp, 449  
    demo/sdhoptions.cpp, 455  
**sdhport**  
  cSDHOptions, 304  
**SDHUSAGE\_DEFAULT**  
  cpp.desire.final/demo/sdhoptions.h, 460  
  demo/sdhoptions.h, 462  
**sdhusage\_dsacom**  
  cpp.desire.final/demo/sdhoptions.cpp, 449  
  demo/sdhoptions.cpp, 455  
**sdhusage\_dsaother**  
  cpp.desire.final/demo/sdhoptions.cpp, 449  
  demo/sdhoptions.cpp, 455  
**sdhusage\_general**  
  cpp.desire.final/demo/sdhoptions.cpp, 450  
  demo/sdhoptions.cpp, 456  
**sdhusage\_sdhcom\_cancommon**  
  cpp.desire.final/demo/sdhoptions.cpp, 450  
  demo/sdhoptions.cpp, 456  
**sdhusage\_sdhcom\_common**  
  cpp.desire.final/demo/sdhoptions.cpp, 450  
  demo/sdhoptions.cpp, 456  
**sdhusage\_sdhcom\_esdcan**  
  cpp.desire.final/demo/sdhoptions.cpp, 451  
  demo/sdhoptions.cpp, 457  
**sdhusage\_sdhcom\_peakcan**  
  cpp.desire.final/demo/sdhoptions.cpp, 451  
  demo/sdhoptions.cpp, 457  
**sdhusage\_sdhcom\_serial**  
  cpp.desire.final/demo/sdhoptions.cpp, 451  
  demo/sdhoptions.cpp, 457  
**sdhusage\_sdhoother**  
  cpp.desire.final/demo/sdhoptions.cpp, 451  
  demo/sdhoptions.cpp, 457  
**selgrip**  
  SDH::cSDHSerial, 321, 331  
**Send**  
  SDH::cSDHSerial, 312, 322  
**senscon\_type**  
  SDH::cDSA::sControllerInfo, 105  
**sensor\_info**  
  SDH::cDSA, 101  
**sensorinfo**  
  cSDHOptions, 304  
**serial\_no**  
  SDH::cDSA::sControllerInfo, 105  
  SDH::cDSA::sSensorInfo, 110  
**SetAxisEnable**  
  SDH::cSDH, 178, 179, 226, 227  
**SetAxisMotorCurrent**  
  SDH::cSDH, 176, 177, 223, 225  
**SetAxisTargetAcceleration**  
  SDH::cSDH, 192, 194, 240, 242  
**SetAxisTargetAngle**  
  SDH::cSDH, 183, 184, 231, 232  
**SetAxisTargetVelocity**  
  SDH::cSDH, 186, 188, 234, 236  
**SetAxisValueVector**

SDH::cSDH, 164, 212  
 SetColor  
     SDH::cDBG, 85  
 SetCondition  
     SDH::cIsGraspedByArea, 126, 127  
 SetController  
     SDH::cSDH, 174, 221  
 SetDebugOutput  
     SDH::cSDH, 164, 212  
     SDH::cSDHBase, 285, 286  
 SetFingerEnable  
     SDH::cSDH, 201, 202, 249, 250  
 SetFingerTargetAngle  
     SDH::cSDH, 203, 204, 251, 252  
 SetFlag  
     SDH::cDBG, 84, 85  
 SetFramerate  
     SDH::cDSA, 96, 99  
 SetFramerateRetries  
     SDH::cDSA, 96, 100  
 SetOutput  
     SDH::cDBG, 85  
 SetTimeout  
     SDH::cCANSerial\_ESD, 59, 60  
     SDH::cCANSerial\_PEAK, 70, 72  
     SDH::cRS232, 137, 139  
     SDH::cSerialBase, 338, 339  
 SetVelocityProfile  
     SDH::cSDH, 175, 223  
 size  
     SDH::cDSA::sResponse, 109  
 SleepSec  
     SDH, 51  
 sn  
     SDH::cSDHSerial, 319, 329  
 soc  
     SDH::cSDHSerial, 320, 330  
 soc\_date  
     SDH::cSDHSerial, 320, 330  
 sResponse  
     SDH::cDSA::sResponse, 108  
 sTactileSensorFrame  
     SDH::cDSA::sTactileSensorFrame, 111  
 start\_dsa  
     SDH::cDSA, 102  
 start\_pc  
     SDH::cDSA, 102  
 state  
     SDH::cSDHSerial, 318, 328  
 status  
     SDH::cCANSerial\_ESD, 61  
     SDH::cRS232, 140  
 status\_flags  
     SDH::cDSA::sControllerInfo, 105  
 Stop  
     SDH::cSDH, 173, 221  
 stop  
     SDH::cSDHSerial, 316, 326  
 StoreNow  
     SDH::cSimpleTime, 350  
 StrDupNew  
     cpp.desire.final/sdh/rs232-cygwin.cpp, 524  
     sdh/rs232-cygwin.cpp, 526  
 sw\_version  
     SDH::cDSA::sControllerInfo, 105  
 SWAP\_FLAGS  
     cpp.desire.final/vcc/getopt.c, 603  
     vcc/getopt.c, 606  
 symbol  
     SDH::cUnitConverter, 365  
 Sync  
     SDH::cSDHSerial, 312, 322  
 SyncUnknown  
     SDH::cSDHSerial, 313, 322  
 tCRCValue  
     SDH, 47  
 temp  
     SDH::cSDHSerial, 319, 329  
 temp\_electronics  
     SDH::cSDHSerial, 319, 329  
 terminal  
     SDH::cSDHSerial, 315, 325  
 texel  
     SDH::cDSA::sTactileSensorFrame, 112  
 texel\_height  
     SDH::cDSA::sMatrixInfo, 107  
 texel\_offset  
     SDH::cDSA, 102  
 texel\_width  
     SDH::cDSA::sMatrixInfo, 107  
 timeout  
     cSDHOpts, 304  
     SDH::cSerialBase, 341  
 timestamp  
     SDH::cDSA::sTactileSensorFrame, 112  
 Timeval  
     SDH::cSimpleTime, 350, 351  
 ToExternal  
     SDH::cUnitConverter, 362–364  
 ToIndexVector  
     SDH::cSDH, 165, 213  
 ToInternal  
     SDH::cUnitConverter, 363, 364  
 ToRange  
     SDH, 51  
 ts  
     SDH::cIsGraspedBase, 122

tTexel  
    SDH::cDSA, 93

uc\_angle  
    SDH::cSDH, 264, 266

uc\_angle\_degrees  
    SDH::cSDH, 260

uc\_angle\_radians  
    SDH::cSDH, 260

uc\_angular\_acceleration  
    SDH::cSDH, 264, 266

uc\_angular\_acceleration\_degrees\_per\_second\_squared  
    SDH::cSDH, 261

uc\_angular\_acceleration\_radians\_per\_second\_squared  
    SDH::cSDH, 261

uc\_angular\_velocity  
    SDH::cSDH, 264, 266

uc\_angular\_velocity\_degrees\_per\_second  
    SDH::cSDH, 260

uc\_angular\_velocity\_radians\_per\_second  
    SDH::cSDH, 260

uc\_identity  
    SDH, 52

uc\_motor\_current  
    SDH::cSDH, 264, 266

uc\_motor\_current\_ampere  
    SDH::cSDH, 261

uc\_motor\_current\_milliampere  
    SDH::cSDH, 261

uc\_position  
    SDH::cSDH, 264, 266

uc\_position\_meter  
    SDH::cSDH, 261

uc\_position\_millimeter  
    SDH::cSDH, 261

uc\_temperature  
    SDH::cSDH, 264, 266

uc\_temperature\_celsius  
    SDH::cSDH, 260

uc\_temperature\_fahrenheit  
    SDH::cSDH, 260

uc\_time  
    SDH::cSDH, 264, 266

uc\_time\_milliseconds  
    SDH::cSDH, 260

uc\_time\_seconds  
    SDH::cSDH, 260

uid  
    SDH::cDSA::sMatrixInfo, 107

UInt16  
    SDH, 47

UInt32

SDH, 47

UInt8  
    SDH, 48

ungetch  
    SDH::cSerialBase, 341

ungetch\_valid  
    SDH::cSerialBase, 341

UpdateFrame  
    SDH::cDSA, 96, 99

usage  
    cpp.desire.final/demo/demo-contact-grasping.cpp, 389  
    cpp.desire.final/demo/demo-dsa.cpp, 395  
    cpp.desire.final/demo/demo-GetAxisActualAngle.cpp, 400  
    cpp.desire.final/demo/demo-GetFingerXYZ.cpp, 407  
    cpp.desire.final/demo/demo-simple-withtiming.cpp, 416  
    cpp.desire.final/demo/demo-simple.cpp, 420  
    cpp.desire.final/demo/demo-velocity-acceleration.cpp, 436  
    cSDHOOptions, 304  
    demo-griphand.cpp, 615  
    demo/demo-contact-grasping.cpp, 392  
    demo/demo-dsa.cpp, 398  
    demo/demo-GetAxisActualAngle.cpp, 403  
    demo/demo-GetFingerXYZ.cpp, 410  
    demo/demo-simple-withtiming.cpp, 418  
    demo/demo-simple.cpp, 422  
    demo/demo-velocity-acceleration.cpp, 438

use\_can\_esd  
    cSDHOOptions, 304

use\_can\_peak  
    cSDHOOptions, 304

use\_fahrenheit  
    cSDHOOptions, 304

USE\_HANDLE  
    cpp.desire.final/sdh/canserial-peak.cpp, 476  
    sdh/canserial-peak.cpp, 478

USE\_HANDLES  
    cpp.desire.final/sdh/canserial-peak.cpp, 476  
    sdh/canserial-peak.cpp, 478

use.radians  
    cSDHOOptions, 304

UseDegrees  
    SDH::cSDH, 166, 214

user\_errors  
    SDH::cSDHSerial, 314, 324

UseRadians  
    SDH::cSDH, 166, 214

USING\_NAMESPACE\_SDH  
    sdhlibrary\_cpp\_sdhlibrary\_settings\_h-derived\_settings\_group, 24

V  
 cpp.desire.final/sdh/dbg.h, 490  
 sdh/dbg.h, 492

v  
 SDH::cSDHSerial, 315, 325

val  
 option, 367

Valid  
 SDH::cSimpleVector, 355

valid  
 SDH::cSimpleVector, 356

value  
 SDH::cSimpleVector, 356

VAR  
 cpp.desire.final/sdh/dbg.h, 490  
 sdh/dbg.h, 492

vcc/getopt.c, 605  
 \_\_\_, 606  
 \_getopt\_initialized, 607  
 \_getopt\_initialize, 607  
 \_getopt\_internal, 607  
 exchange, 607  
 first\_nonopt, 607  
 getopt, 607  
 GETOPT\_INTERFACE\_VERSION, 606  
 last\_nonopt, 607  
 my\_index, 607  
 nextchar, 607  
 NONOPTION\_P, 606  
 optarg, 607  
 opterr, 607  
 optind, 607  
 optopt, 607  
 ordering, 607  
 PERMUTE, 606  
 posixly\_correct, 607  
 REQUIRE\_ORDER, 606  
 RETURN\_IN\_ORDER, 606  
 SWAP\_FLAGS, 606

vcc/getopt.h, 610  
 \_GETOPT\_H, 611  
 \_getopt\_internal, 611  
 getopt, 611  
 getopt\_long, 611  
 getopt\_long\_only, 611  
 no\_argument, 611  
 optarg, 611  
 opterr, 611  
 optind, 611  
 optional\_argument, 611  
 optopt, 611  
 required\_argument, 611

vcc/getopt1.c, 613  
 GETOPT\_INTERFACE\_VERSION, 613

getopt\_long, 613  
 getopt\_long\_only, 613  
 NULL, 613

vel  
 SDH::cSDHSerial, 318, 328

ver  
 SDH::cSDHSerial, 319, 329

ver\_date  
 SDH::cSDHSerial, 319, 329

vlim  
 SDH::cSDHSerial, 315, 325

vp  
 SDH::cSDHSerial, 316, 326

WaitAxis  
 SDH::cSDH, 181, 183, 229, 231

what  
 SDH::cSDHLibraryException, 299

write  
 SDH::cCANSerial\_ESD, 58, 60  
 SDH::cCANSerial\_PEAK, 70, 71  
 SDH::cRS232, 136–139  
 SDH::cSerialBase, 338, 340

WriteCommand  
 SDH::cDSA, 94, 98

WriteCommandWithPayload  
 SDH::cDSA, 94, 97

x  
 SDH::cSimpleVector, 354, 355

y  
 SDH::cSimpleVector, 355

z  
 SDH::cSimpleVector, 355

zeros\_v  
 SDH::cSDH, 262, 265