SDHLibrary-python

0.0.2.9

Generated by Doxygen 1.7.3-20110123

Tue Sep 30 2014 15:53:34

Contents

Chapter 1

SDHLibrary_python

The python package (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** package ("library") for the python scripting language.

- For an overview of the library itself see: the sdh package.
- For some demonstration scripts see: demonstration scripts.

1.3 Links

- The SCHUNK homepage: `http://www.schunk.com/`
- The python homepage: `http://www.python.org/`
- This package has been tested and used with python versions 2.5 and 2.6.

- Additionally used python packages (mostly included on the CD):
  - The **pySerial** module: http://pyserial.sourceforge.net/
  - For some demo scripts with graphical output **Tkinter** is used. This is usually included in a python distribution
  - Windows specific:
    - Python windows compatibility package (stdlib) **pywin32:** http://sourceforge.net/pr
    - Python package for emulating readline **readline.py** http://newcenturycomputers.net
  - **py.test** unit testing framework from http://codespeak.net/py/current/doc/index.h
- The documentation of **doxygen** (documentation generator) can be found:
  - On the web: http://www.stack.nl/∼dimitri/doxygen/
- For extracting native python docstrings containing doxygen markup the doxypy.py filter is used, see http://code.foosel.org/doxypy

1.4 Copyright

Chapter 2

Bug List

Class sdh.dsa.cDSA SCHUNK-internal bugzilla ID: Bug 983

With SDHLibrary-Python < 0.0.2.1 communication to the DSACON32m tactile sensor controller within the SDH was not established correctly in some cases. The default baudrate of the RS232 port was not set correctly unless specified explicitly.

=> Resolved in SDHLibrary-Python 0.0.2.1

Member sdh.dsa.cDSA.GetMatrixSensitivity With DSACON32m firmware R218 and before this did not work, instead the factory default (0.5) was always reported

=> Resolved in DSACON32m firmware R268

Member sdh.dsa.cDSA.SetFramerate SCHUNK-internal bugzilla ID: Bug 680

With DSACON32m firmware R276 and after and SDHLibrary-Python v0.0.1.19 and before stopping of the sending did not work.

=> Resolved in SDHLibrary-Python v0.0.1.20

SCHUNK-internal bugzilla ID: Bug 680

With DSACON32m firmware before R276 and SDHLibrary-Python v0.0.1.20 and before acquiring of single frames did not work

=> Resolved in SDHLibrary-Python v0.0.1.21

SCHUNK-internal bugzilla ID: Bug 703

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

=> Resolved in DSACON32m firmware 2.9.0.0

=> Resolved in SDHLibrary-Python v0.0.2.1 with workaround for older DSACON32m firmwares

Member sdh.dsa.cDSA.SetFramerateRetries With DSACON32m firmware R276 and after and SDHLibrary-Python v0.0.1.19 and before stopping of the sending did not work.

=> Resolved in SDHLibrary-Python v0.0.1.20

With DSACON32m firmware before R276 and SDHLibrary-Python v0.0.1.20 and before acquiring of single frames did not work

=> Resolved in SDHLibrary-C++ v0.0.1.21

Member sdh.sdh.cSDH.FastStop For now this will **NOT** work while a GripHand() command is executing, even if that was initiated non-sequentially!

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

=> Resolved in SDH firmware 0.0.2.6

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

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 fast stopped.

Member sdh.sdh.cSDH.MoveAxis 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.sdh.cSDH.MoveFinger 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.sdh.cSDH.MoveHand 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.

=> Resolved in SDH firmware 0.0.2.7

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

With SDH firmware < 0.0.2.7 this made the axis jerk in eCT_POSE controller type.

=> Resolved in SDH firmware 0.0.2.7

With SDH firmware < 0.0.2.7 this made the axis jerk in eCT_POSE controller type.

=> Resolved in SDH firmware 0.0.2.7

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

=> Resolved in SDH firmware 0.0.2.6

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

=> Resolved in SDH firmware 0.0.2.7

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

=> Resolved in SDH firmware 0.0.2.7

Class sdh.sdhserial.cSDHSerial SCHUNK-internal bugzilla ID: Bug 1517

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

=> Resolved in SDHLibrary-python 0.0.2.8

Member sdh.sdhserial.cSDHSerial.kv 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.sdhserial.cSDHSerial.pid 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::auxiliary.GetAvailablePorts SCHUNK-internal bugzilla ID: Bug 1013

On Linux the probing for available ports seems to disturb communication to already opened ports. This inhibits demo-gui.py from working.

=> Resolved in SDHLibrary-Python 0.0.2.2

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 5

Class Index

5.1 Class Hierarchy

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

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Module Documentation

8.1 Demonstration scripts

Some demonstration scripts are provided which show the usage of the sdh python package:

Files

- file demo-benchmark.py

  Simple script to do grasping using tactile sensor info feedback. See demo-benchmark.__doc__ and the online help ("-h" or "--help") for a list of available options.

- file demo-calc-workspace.py

  Output a data file with xyz fingertip positions for all possible angles.

- file demo-contact-grasping.py

  Simple script to do grasping using tactile sensor info feedback. See demo-contact-grasping.__doc__ and the online help ("-h" or "--help") for a list of available options.

- file demo-dsa.py

  Simple script to access tactile sensors of SDH. See demo-dsa.__doc__ and online help ("-h" or "--help") for available options.

- file demo-GetAxisActualAngle.py

  Print current actual axis angles from SDH. See demo-GetAxisActualAngle.__doc__ and online help ("-h" or "--help") for available options.

- file demo-gui.py

  Simple GUI (Graphical User Interface) to control an SDH. See demo-gui.__doc__ and online help ("-h" or "--help") for available options.

- file demo-radians.py

Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control).

- file demo-simple.py

  Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control).

- file demo-simple2.py

  Very simple demonstration of the sdh python package.

- file demo-simple3.py

  Very simple demonstration of the sdh python package: Make the SDH move 3 axes.

- file demo-tactile.py

  Simple GUI to visualize tactile sensors of SDH. See demo-tactile.__doc__ and the online help ("-h" or "--help") for available options.

- file demo-temperature.py

  Print measured temperatures of SDH. See demo-temperature.__doc__ and online help ("-h" or "--help") for available options.

- file demo-velocity-acceleration.py

  Demonstration script of the sdh python package: Make the SDH move one finger in "velocity with acceleration ramp" control mode.

- file demo-workspace.py

  Move fingers to show workspace of SDH. (Python demo script using the sdh.py import library.)

- file sdh-ping.py

  Measure response time of SDH See sdh-ping.__doc__ and online help ("-h" or "--help") for available options.

8.1.1 Detailed Description

Some demonstration scripts are provided which show the usage of the sdh python package:

8.2 Online help of demonstration scripts

The provided demonstration scripts have an online help which is shown below:

The provided demonstration scripts have an online help which is shown below:

Online help for script `demo-benchmark.py`

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

- Example usage:
  - Start moving wildly using an SDH that is connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-benchmark.py --tcp=192.168.1.42:23


  - Start moving wildly using an SDH that is connected to:
    - port 2 = COM3 (joint controllers) and
    > demo-benchmark.py -p 2


  - Start moving wildly using an SDH that is connected to:
    - USB to RS232 converter 0 (joint controllers) and
    > demo-benchmark.py --sdh_rs_device=/dev/ttyUSB0


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


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

usage: demo-benchmark.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
```

```
-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
-v, --version         Print the version (revision/release names) of script,
                      library, python interpreter (PC-side) and firmware
                      release, date (SDH-side) then exit.
-V, --version_check   Check the firmware release of the connected SDH if it
                      is the one recommended by this library. A message will
                      be printed accordingly.
-T TIMEOUT, --timeout=TIMEOUT
                      Timeout in seconds (float accepted) used to wait for
                      answers from SDH (default is None = wait for ever).
-c, --can             use the (ESD) CAN interface instead of RS232.
                      (Requires the windows python.exe not the cygwin one)
-n NET, --net=NET     use the ESD CAN net number NET
--id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                      SDH. Default is 43.
--id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                      SDH. Default is 42.
--baudrate=BAUDRATE, --baud=BAUDRATE
                      use the BAUDRATE for communication. Default is 115200
                      Bit/s for RS232 and 1 MBit/s for CAN.
--tcp=[IP_OR_HOSTNAME][:PORT]
                      use TCP for communication with the SDH. The SDH can be
                      reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                      which can be a numeric IPv4 address or a hostname. The
                      default is 192.168.1.42:23 (to use the default you
                      have to specify '--tcp='). When using --tcp and
                      --dsa_tcp then only the last set IP_OR_HOSTNAME is
                      used for both. (This feature requires at least SDH
                      firmware 0.0.3.1)
--dsaport=PORT        use serial communication port PORT to connect to DSA
                      (tactile sensor of SDH) instead of default
                      4='COM5'='/dev/ttyS4'.
--dsa_rs_device=DEVICE_FORMAT_STRING
                      Use DEVICE_FORMAT_STRING instead of the default
                      "/dev/ttyS%d". Useful on Linux, 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.
--dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                      use TCP for communication with the DSA (tactile sensor
                      of SDH). The DSA can be reached via TCP/IP on port
                      PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                      address or a hostname. The default is
                      192.168.1.42:13000 (to use the default you have to
                      specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                      then only the last set IP_OR_HOSTNAME is used for
                      both. (This feature requires at least SDH firmware
                      0.0.3.2)
--dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                      (tactile sensor of SDH). The default is 13000. (This
                      feature requires at least SDH firmware 0.0.3.2)
-r FRAMERATE, --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)).
```

```
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script demo-calc-workspace.py

```
Usage:
Output a data file with xyz fingertip positions for all possible angles

usage: demo-calc-workspace.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
```

```
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  --s0=STEP             Set step width for finger axis angle 0 to STEP,
                        default=5.
  --s1=STEP             Set step width for finger axis angle 1 (axis angles
                        1/3/5) to STEP, default=5.
  --s2=STEP             Set step width for finger axis angle 2 (axis angles
                        2/4/6) to STEP, default=5.
  -t TYPE, --type=TYPE  The type of points to generate: 'all' : full
                        workspace, 'surface' only the surface
```

Online help for script `demo-contact-grasping.py`

```
Usage: 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:
  - Start grasping using an SDH that is connected via Ethernet:
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The joint controller is attached to the
    TCP port 23 and the tactile sensors to TCP port 13000 (default).
    (Requires at least SDH-firmware v0.0.3.2)
```

```
    > demo-contact-grasping.py --tcp=192.168.1.42:23 --dsa_tcp=


  - Start grasping using an SDH that is connected to:
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-contact-grasping.py -p 2 --dsaport=3


  - Start grasping using an SDH that is connected to:
    - USB to RS232 converter 0 (joint controllers) and
    - USB to RS232 converter 1 (tactile sensor controller)
    > demo-contact-grasping.py --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/
      ttyUSB1


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


  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected to
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-contact-grasping.py -p 2 --dsaport=3 -v

usage: demo-contact-grasping.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
```

```
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  --desired-force=DESIRED_FORCE
                        Desired force that every tactile sensor should reach,
                        default = 5.0.
  --vel-per-force=VEL_PER_FORCE
                        Desired velocity per force factor, default = 1.25.
  --dsadebug=LEVEL      Print debug messages of the cDSA object of level LEVEL
                        or lower while processing the python script. Level 0
                        (default): No messages,  1: cDSA-level messages, 2:
                        cSDHSerial-level messages
  --dsadebuglog=LOGFILE
                        Redirect the printed dsa debug messages to LOGFILE
                        instead of standard error (default). If LOGFILE starts
                        with '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
```

Online help for script demo-dsa.py

```
Usage:
Simple script to demonstrate the use of the sdh.dsa module and the
sdh.dsa.cDSA class in the sdh package.

Remarks:
- You must specify at least one of these options to see some output:
  -f | --fullframe
  -C | --resulting
  -c | --controllerinfo
  -s | --sensorinfoinfo
  -m | --matrixinfo=N

- Example usage:
  - Read a single full frame from tactile sensors connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The tactile sensors use TCP port 13000 (default).
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-dsa.py --dsa_tcp=192.168.1.42 -f

  - Read a single full frame from tactile sensors connected to port 3 = COM4:
    > demo-dsa.py --dsaport=3 -f

  - Read a single full frame from tactile sensors connected to USB to
    RS232 converter 0:
    > demo-dsa.py --dsa_rs_device=0 -f

  - Read full frames continuously from tactile sensors connected to port 3 = COM4
      :
    > demo-dsa.py --dsaport=3 -f -r 1

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

  - Read full frames continuously as fast as possible (DSA push-mode)
    from tactile sensors connected to port 3 = COM4:
    > demo-dsa --dsaport=3 -f -r 30

  - Read resulting values (contact area, contact force) continuously
    from tactile sensors connected to port 3 = COM4:
    > demo-dsa.py --dsaport=3 -C -r 1

  - Read the sensor, controller, matrix 0 and matrix 1 infos
    from tactile sensors connected to port 3 = COM4:
    > demo-dsa.py --dsaport=3 -s -c -m 0 -m 1

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

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

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

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

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

  - Reset the sensitivity of all tactile sensor matrixes to their factory default
      .
    The value will be used only temporarily (until reset or power cycle).
    > demo-dsa.py --dsaport=3 --sensitivity=0.75 --reset

  - Set the sensitivity of tactile sensor matrices 1 and 4 to individual
"
    values temporarily.
"
    The value will be used only temporarily (until reset or power cycle).
"
    Sensor 1 (distal sensor of finger 1) will be set to 0.1
"
    Sensor 4 (proximal sensor of finger 3) will be set to 0.4
"
    > demo-dsa.py --dsaport=3 --sensitivity1=0.1 --sensitivity4=0.4

  - Like for the sensitivity the threshold can be adjusted using
    the --threshold=VALUE or --thresholdX=VALUE arguments.

usage: demo-dsa.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
```

```
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  -f, --fullframe       Print acquired full frames numerically.
  -C, --resulting       Print calculated resulting values (area, force).
  -s, --sensorinfo      Print sensor info from DSA (texel dimensions, number
                        of texels...).
  -c, --controllerinfo  Print controller info from DSA (version...).
  -m MATRIX_INDEX, --matrixinfo=MATRIX_INDEX
                        Print matrix info for matrix with index MATRIX_INDEX
                        from DSA. This option can be used multiple times.
  --calibration         Calibrate voltage to pressure calculation.
  --calib_pressure=CALIB_PRESSURE
                        Pressure calibration value. E.g. determined using the
                        --calibration option.
  --calib_voltage=CALIB_VOLTAGE
                        Voltage calibration value. E.g. determined using the
                        --calibration option.
  --calibration-force=CALIBRATION_FORCE
                        External force applied for calibration in N (weight in
```

```
                      kg * 9.81).
--sensitivity=SENSITIVITY
                      Set the sensor sensitivities for all tactile sensor
                      pads                              to the given value
                      [0.0 .. 1.0] (0.0 is minimum, 1.0
                      is maximum sensitivity).
                      If --reset is given as well then SENSITIVITY is
                      ignored and                              the
                      sensitivities are reset to the factory defaults.
                      To see the current setting for sensitivity use
                      --showdsainfo.
                      For setting sensitivities individually for a specific
                      sensor X [0..5] use --sensitivityX=SENSITIVITY
--sensitivity0=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 0.
--sensitivity1=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 1.
--sensitivity2=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 2.
--sensitivity3=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 3.
--sensitivity4=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 4.
--sensitivity5=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 5.
--threshold=THRESHOLD
                      Set the sensor threshold for all tactile sensor pads
                      to the given value [0.0 .. 1.0] (0.0 is minimum, 1.0
                      is maximum threshold).
                      If --reset is given as well then THRESHOLD is ignored
                      and the thresholds are reset to the factory defaults.
                      To see the current setting for threshold use
                      --showdsainfo.
                      For setting thresholds individually for a specific
                      sensor X [0..5] use --thresholdX=THRESHOLD
--threshold0=THRESHOLD
                      Set sensor threshold specifically for sensor 0.
--threshold1=THRESHOLD
                      Set sensor threshold specifically for sensor 1.
--threshold2=THRESHOLD
                      Set sensor threshold specifically for sensor 2.
--threshold3=THRESHOLD
                      Set sensor threshold specifically for sensor 3.
--threshold4=THRESHOLD
                      Set sensor threshold specifically for sensor 4.
--threshold5=THRESHOLD
                      Set sensor threshold specifically for sensor 5.
--reset               If given, then the values given with --sensitivity(X)
                      and/or --threshold(X) are reset to their factory
                      default.
--persistent          If given then all the currently set values for the
                      sensitivity or threshold are saved persistently
                      in the configuration memory of the DSACON32m
                      controller in the SDH.
                      PLEASE NOTE: the maximum write endurance of the
                      configuration memory is about 100.000 times!
--showdsasettings     If given, then current settings for sensitivity and
                      threshold will be printed on stdout first.
```

Online help for script `demo-GetAxisActualAngle.py`

```
Usage:
Print current actual axis angles from SDH.

- Example usage:
  - Print actual angles of an SDH connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-GetAxisActualAngle.py --tcp=192.168.1.42:23

  - Print actual angles of an SDH connected to port 2 = COM3 once:
    > demo-GetAxisActualAngle.py -p 2

  - Print actual angles of an SDH connected to port 2 = COM3 every 500ms:
    > demo-GetAxisActualAngle.py -p 2 -t 0.5

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

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

usage: demo-GetAxisActualAngle.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
```

```
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  -t PERIOD, --period=PERIOD
                        Time period of measurements in seconds. The default of
                        '0' means: report once only.
  -k, --xyz             Enable calculation of forward kinematics.
  -s, --get_actual_velocity
                        Flag, if given then the actual velocity is read from
                        the SDH too.
```

```
  -P VELOCITY_PROFILE, --velocity_profile=VELOCITY_PROFILE
                        Id of the velocity profile to use (0=sin
                        square(default), 1=ramp)
  -m MOVE, --move=MOVE  Target positions for a SetAxisTargetAngle command. If
                        given the fingers are moved there while reporting
                        Actual axis angles. Example:
                        '0,10,2.0,-30,-44,-55,-66'.
  -w VELOCITY, --velocity=VELOCITY
                        Target velocities for a SetAxisTargetVelocity command.
                        If given the fingers are moved with the given max
                        velocities while reporting Actual axis angles.
                        Example: '28,40,40,40,40,40,40'.
  -a ACCELERATION, --acceleration=ACCELERATION
                        Target accelerations for a SetAxisTargetAcceleration
                        command. If given the fingers are moved with the given
                        max accelerations while reporting Actual axis angles.
                        Example: '100,100,100,100,100,100,100'.
```

Online help for script `demo-gui.py`

```
Usage:
Simple GUI (Graphical User Interface) to command an SDH.

A window based interactive application with GUI (Graphical User
Interface) is started. It allows to command the SDH interactively
with mouse and keyboard. The application uses the sdh.py python import
library to connect to the SDH and the Tkinter package (Tk for python)
to build the gui elements.
If no RS232 port or CAN options to select the communication channel are
given on the command line then an interactive window to select the
channel is shown first before the actual program is started.

Apart from the intuitive gui elements the following keyboard shortcuts
can be used:
- <Pause/Break> : FastStop
- <CTRL-c>      : Close application after powering off the controllers
- <CTRL-C>      : Close application and keep controllers powered
- <CTRL-m>      : Move axes of hand to the values set with the sliders
- <CTRL-s>      : Stop movement of axes, but keep controllers enabled
- <CTRL-a>      : Set Sliders to the current actual angle of the corresponding
                  axis
- <CTRL-h>      : Set target position to "home" (all 0.0) Warning: collisions
                  are not checked!

- Example usage:
  - Start interactive GUI. Before the actual interface appears a
    window will appear to interactively select the interface to use.
    > demo-gui.py

  - Start interactive GUI using an SDH connected via Ethernet.
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-gui.py --tcp=192.168.1.42

  - Start interactive GUI using an SDH connected to port 2 = COM3.
    (The port for the tactile sensors can be choosen interactively)
    > demo-gui.py -p 2

  - 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-gui.py -p 2 --dsaport=3 -v

usage: demo-gui.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
```

```
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  -f FILE, --file=FILE  Load positions from file FILE.
  -r FRAMERATE, --framerate=FRAMERATE
                        Framerate for updating the tactile sensor display. The
                        DSACON32m controller in the SDH will always send data
                        at the highest possible rate (ca. 30 FPS (frames per
                        second)). The actual reachable rate of updates depends
                        on your system (CPU/memory).
  -s STYLE, --style=STYLE
                        Display style. Valid styles are: ['color', 'grey',
                        'dec', 'percent']. This option can be given multiple
                        times. So to set grey and percent use '-s grey -s
                        percent'.
  --ivfile=FILE         Use FILE as iv_filename parameter to
                        cSDH.CheckFingerCollisions().
  --sensitivity=SENSITIVITY
                        Set the sensor sensitivities for all tactile sensor
                        pads                              to the given value
                        [0.0 .. 1.0] (0.0 is minimum, 1.0
                        is maximum sensitivity).
                        If --reset is given as well then SENSITIVITY is
                        ignored and                                 the
                        sensitivities are reset to the factory defaults.
                        To see the current setting for sensitivity use
                        --showdsasettings.
                        For setting sensitivities individually for a specific
                        sensor X [0..5] use --sensitivityX=SENSITIVITY
  --sensitivity0=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 0.
  --sensitivity1=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 1.
  --sensitivity2=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 2.
  --sensitivity3=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 3.
  --sensitivity4=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 4.
  --sensitivity5=SENSITIVITY
                        Set sensor sensitivity specifically for sensor 5.
```

```
  --threshold=THRESHOLD
                        Set the sensor threshold for all tactile sensor pads
                        to the given value [0.0 .. 1.0] (0.0 is minimum, 1.0
                        is maximum threshold).
                        If --reset is given as well then THRESHOLD is ignored
                        and the thresholds are reset to the factory defaults.
                        To see the current setting for threshold use
                        --showdsasettings.
                        For setting thresholds individually for a specific
                        sensor X [0..5] use --thresholdX=THRESHOLD
  --threshold0=THRESHOLD
                        Set sensor threshold specifically for sensor 0.
  --threshold1=THRESHOLD
                        Set sensor threshold specifically for sensor 1.
  --threshold2=THRESHOLD
                        Set sensor threshold specifically for sensor 2.
  --threshold3=THRESHOLD
                        Set sensor threshold specifically for sensor 3.
  --threshold4=THRESHOLD
                        Set sensor threshold specifically for sensor 4.
  --threshold5=THRESHOLD
                        Set sensor threshold specifically for sensor 5.
  --reset               If given, then the values given with --sensitivity(X)
                        and/or --threshold(X) are reset to their factory
                        default.
  --persistent          If given then all the currently set values for the
                        sensitivity or threshold are saved persistently
                        in the configuration memory of the DSACON32m
                        controller in the SDH.
                        PLEASE NOTE: the maximum write endurance of the
                        configuration memory is about 100.000 times!
  --showdsasettings     If given, then current settings for sensitivity and
                        threshold will be printed on stdout first.
```

Online help for script `demo-radians.py`

```
Usage:
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10
using the "pose" (coordinated position) control mode.

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

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

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

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

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

usage: demo-radians.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
```

```
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script `demo-simple`.py

```
Usage:
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10
using the "pose" (coordinated position) control mode.

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

  - Make SDH connected to port 2 = COM3 move:
    > demo-simple.py -p 2

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

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

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

usage: demo-simple.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
```

```
                          used for both. (This feature requires at least SDH
                          firmware 0.0.3.1)
  --dsaport=PORT          use serial communication port PORT to connect to DSA
                          (tactile sensor of SDH) instead of default
                          4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                          Use DEVICE_FORMAT_STRING instead of the default
                          "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                          use TCP for communication with the DSA (tactile sensor
                          of SDH). The DSA can be reached via TCP/IP on port
                          PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                          address or a hostname. The default is
                          192.168.1.42:13000 (to use the default you have to
                          specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                          then only the last set IP_OR_HOSTNAME is used for
                          both. (This feature requires at least SDH firmware
                          0.0.3.2)
  --dsa_tcp_port=PORT     use TCP port PORT for communication with the DSA
                          (tactile sensor of SDH). The default is 13000. (This
                          feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle                Disable RLE (Run Length Encoding) for acquiring full
                          frames.
```

Online help for script `demo-simple2.py`

```
Usage:
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10.
The movement will be stopped in the middle of the movement.

- Example usage:
  - Make SDH connected connected via Ethernet move:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-simple2.py --tcp=192.168.1.42:23

  - Make SDH connected to port 2 = COM3 move:
    > demo-simple2.py -p 2

  - Make SDH connected to USB to RS232 converter 0 move:
    > demo-simple2.py --sdh_rs_device=/dev/ttyUSB0

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The joint controller is attached to the
    TCP port 23 and the tactile sensors to TCP port 13000.
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-simple2.py --tcp=192.168.1.42:23 --dsa_tcp=:13000 -v
```

```
  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected to:
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-simple2.py --port=2 --dsaport=3 -v

usage: demo-simple2.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
```

```
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script `demo-simple3.py`

```
Usage:
Very simple python demo script using the sdh package.
Will move axes 1,2 and 3 to a fixed position, then
return back to the home position.
The movement will be started 'non sequentially' and
the scripts then waits for the movement to be finished.

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

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

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

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The joint controller is attached to the
    TCP port 23 and the tactile sensors to TCP port 13000.
```

```
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-simple3.py --tcp=192.168.1.42:23 --dsa_tcp=:13000 -v

- Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected to:
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-simple3.py --port=2 --dsaport=3 -v

usage: demo-simple3.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
```

```
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script `demo-tactile`.py

```
Usage:
Simple GUI to visualize tactile sensors of SDH. (Python demo script using the sdh
      .py import library.)

- Example usage:
  - Display the tactile sensors connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The tactile sensors use TCP port 13000 (default).
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-tactile.py --dsa_tcp=192.168.1.42

  - Display the tactile sensors connected to port 3 = COM4
    with the default style (colors, no numbers):
    > demo-tactile.py --dsaport=3

  - Display the tactile sensors connected to USB to RS232
    converter 0 with the default style (colors, no numbers):
    > demo-tactile.py --dsa_rs_device=/dev/ttyUSB0

  - Display the tactile sensors connected to port 3 = COM4
    with greycodes and numerical output in percent:
    > demo-tactile.py --dsaport=3 --style=grey --style=percent
```

```
  - 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-tactile.py -p 2 --dsaport=3 -v

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

usage: demo-tactile.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
```

```
                      4='COM5'='/dev/ttyS4'.
--dsa_rs_device=DEVICE_FORMAT_STRING
                      Use DEVICE_FORMAT_STRING instead of the default
                      "/dev/ttyS%d". Useful on Linux, 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.
--dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                      use TCP for communication with the DSA (tactile sensor
                      of SDH). The DSA can be reached via TCP/IP on port
                      PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                      address or a hostname. The default is
                      192.168.1.42:13000 (to use the default you have to
                      specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                      then only the last set IP_OR_HOSTNAME is used for
                      both. (This feature requires at least SDH firmware
                      0.0.3.2)
--dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                      (tactile sensor of SDH). The default is 13000. (This
                      feature requires at least SDH firmware 0.0.3.2)
--no_rle              Disable RLE (Run Length Encoding) for acquiring full
                      frames.
-r FRAMERATE, --framerate=FRAMERATE
                      Framerate for updating the display. The DSACON32m
                      controller in the SDH will always send data at the
                      highest possible rate (ca. 30 FPS (frames per
                      second)). The actual reachable rate of updates depends
                      on your system (CPU/memory).
-s STYLE, --style=STYLE
                      Display style. Valid styles are: ['color', 'grey',
                      'dec', 'percent']. This option can be given multiple
                      times. So to set grey and percent use '-s grey -s
                      percent'.
--sensitivity=SENSITIVITY
                      Set the sensor sensitivities for all tactile sensor
                      pads                                to the given value
                      [0.0 .. 1.0] (0.0 is minimum, 1.0
                      is maximum sensitivity).
                      If --reset is given as well then SENSITIVITY is
                      ignored and                                  the
                      sensitivities are reset to the factory defaults.
                      To see the current setting for sensitivity use
                      --showdsasettings.
                      For setting sensitivities individually for a specific
                      sensor X [0..5] use --sensitivityX=SENSITIVITY
--sensitivity0=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 0.
--sensitivity1=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 1.
--sensitivity2=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 2.
--sensitivity3=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 3.
--sensitivity4=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 4.
--sensitivity5=SENSITIVITY
                      Set sensor sensitivity specifically for sensor 5.
--threshold=THRESHOLD
                      Set the sensor threshold for all tactile sensor pads
                      to the given value [0.0 .. 1.0] (0.0 is minimum, 1.0
                      is maximum threshold).
```

```
                        If --reset is given as well then THRESHOLD is ignored
                        and the thresholds are reset to the factory defaults.
                        To see the current setting for threshold use
                        --showdsasettings.
                        For setting thresholds individually for a specific
                        sensor X [0..5] use --thresholdX=THRESHOLD
  --threshold0=THRESHOLD
                        Set sensor threshold specifically for sensor 0.
  --threshold1=THRESHOLD
                        Set sensor threshold specifically for sensor 1.
  --threshold2=THRESHOLD
                        Set sensor threshold specifically for sensor 2.
  --threshold3=THRESHOLD
                        Set sensor threshold specifically for sensor 3.
  --threshold4=THRESHOLD
                        Set sensor threshold specifically for sensor 4.
  --threshold5=THRESHOLD
                        Set sensor threshold specifically for sensor 5.
  --reset               If given, then the values given with --sensitivity(X)
                        and/or --threshold(X) are reset to their factory
                        default.
  --persistent          If given then all the currently set values for the
                        sensitivity or threshold are saved persistently
                        in the configuration memory of the DSACON32m
                        controller in the SDH.
                        PLEASE NOTE: the maximum write endurance of the
                        configuration memory is about 100.000 times!
  --showdsasettings     If given, then current settings for sensitivity and
                        threshold will be printed on stdout first.
```

Online help for script `demo-temperature.py`

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

- Example usage:
  - Print temperatures of an SDH connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-temperature.py --tcp=192.168.1.42:23

  - Print temperatures of an SDH connected to port 2 = COM3 once:
    > demo-temperature.py -p 2

  - Print temperatures of an SDH connected to USB to RS232 converter 0 once:
    > demo-temperature.py --sdh_rs_device=/dev/ttyUSB0

  - Print temperatures of an SDH connected to port 2 = COM3 every 500ms:
    > demo-temperature.py -p 2 -t 0.5

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The joint controller is attached to the
    TCP port 23 and the tactile sensors to TCP port 13000.
```

```
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-temperature.py --tcp=192.168.1.42:23 --dsa_tcp=:13000 -v

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

usage: demo-temperature.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
```

```
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
  -t PERIOD, --period=PERIOD
                        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 beginning of
                        every line
```

Online help for script `demo-velocity-acceleration.py`

```
Usage:
Very simple python demo script using the sdh package.
Will move first finger in "velocity with acceleration ramp" control mode.

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

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

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

  - Get the version info of both the joint controllers and the tactile
```

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

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

usage: demo-velocity-acceleration.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
```

```
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script `demo-workspace.py`

```
Usage:
Move fingers to show workspace of SDH. (Python demo script using the sdh.py impor
      t library.)

usage: demo-workspace.py [options]

Options:
  -h, --help            show this help message and exit
  -p PORT, --port=PORT  use serial communication port PORT to connect to SDH
                        instead of default 0='COM1'='/dev/ttyS0'.
  --sdh_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  -d LEVEL, --debug=LEVEL
```

```
                        Print debug messages of level LEVEL or lower while
                        processing the python script. Level 0 (default): No
                        messages,  1: Script-level messages, 2: cSDH-level
                        messages, 3: cSDHSerial-level messages
  -l LOGFILE, --debuglog=LOGFILE
                        Redirect the printed debug messages to LOGFILE instead
                        of standard error (default). If LOGFILE starts with
                        '+' then the output will be appended to the file
                        (without the leading '+'), else the file will be
                        rewritten.
  -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
  -v, --version         Print the version (revision/release names) of script,
                        library, python interpreter (PC-side) and firmware
                        release, date (SDH-side) then exit.
  -V, --version_check   Check the firmware release of the connected SDH if it
                        is the one recommended by this library. A message will
                        be printed accordingly.
  -T TIMEOUT, --timeout=TIMEOUT
                        Timeout in seconds (float accepted) used to wait for
                        answers from SDH (default is None = wait for ever).
  -c, --can             use the (ESD) CAN interface instead of RS232.
                        (Requires the windows python.exe not the cygwin one)
  -n NET, --net=NET     use the ESD CAN net number NET
  --id_read=ID_READ     use the CAN ID ID_READ for receiving messages from the
                        SDH. Default is 43.
  --id_write=ID_WRITE   use the CAN ID ID_WRITE for sending messages to the
                        SDH. Default is 42.
  --baudrate=BAUDRATE, --baud=BAUDRATE
                        use the BAUDRATE for communication. Default is 115200
                        Bit/s for RS232 and 1 MBit/s for CAN.
  --tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the SDH. The SDH can be
                        reached via TCP/IP on port PORT at IP_OR_HOSTNAME,
                        which can be a numeric IPv4 address or a hostname. The
                        default is 192.168.1.42:23 (to use the default you
                        have to specify '--tcp='). When using --tcp and
                        --dsa_tcp then only the last set IP_OR_HOSTNAME is
                        used for both. (This feature requires at least SDH
                        firmware 0.0.3.1)
  --dsaport=PORT        use serial communication port PORT to connect to DSA
                        (tactile sensor of SDH) instead of default
                        4='COM5'='/dev/ttyS4'.
  --dsa_rs_device=DEVICE_FORMAT_STRING
                        Use DEVICE_FORMAT_STRING instead of the default
                        "/dev/ttyS%d". Useful on Linux, 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.
  --dsa_tcp=[IP_OR_HOSTNAME][:PORT]
                        use TCP for communication with the DSA (tactile sensor
                        of SDH). The DSA can be reached via TCP/IP on port
                        PORT at IP_OR_HOSTNAME, which can be a numeric IPv4
                        address or a hostname. The default is
                        192.168.1.42:13000 (to use the default you have to
                        specify '--dsa_tcp='). When using --tcp and --dsa_tcp
                        then only the last set IP_OR_HOSTNAME is used for
                        both. (This feature requires at least SDH firmware
```

```
                        0.0.3.2)
  --dsa_tcp_port=PORT   use TCP port PORT for communication with the DSA
                        (tactile sensor of SDH). The default is 13000. (This
                        feature requires at least SDH firmware 0.0.3.2)
  -r FRAMERATE, --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)).
  --no_rle              Disable RLE (Run Length Encoding) for acquiring full
                        frames.
```

Online help for script miniterm.py

```
USAGE: demo/miniterm.py [options]
    Miniterm - A simple terminal program for the serial port.

    options:
    -p, --port=PORT: RS232 port to use for communication
                     either a number, default = 0 (=COM1 / /dev/ttyS0)
                     or a device name like "/dev/ttyUSB0"
    -b, --baud=BAUD: baudrate, default 115200 for RS232, 1MBit for CAN
    -r, --rtscts:    enable RTS/CTS flow control (default off)
    -e, --echo:      enable local echo (default off)
    -D, --debug:     debug received data (escape nonprintable chars)
    -n, --numeric:       enable numeric mode, see online help (F1+RETURN)
    -x, --hexnumeric:    enable hex numeric mode, see online help (F1+RETURN)
    -a, --additional_ascii enable additional ascii display, see online help (F1+R
      ETURN)
    -c, --cr:        do not send CR+LF, send CR only
    -t, --timeout=TIMEOUT: use timeout of TIMEOUT seconds, default is 0.1
    --xonxoff:   enable software flow control (default off)
    --newline:   do not send CR+LF, send LF only
    --input=FILE: send data from FILE instead of from keyboard
    --nonewline: do not add newlines, mainly usefull when using --input and --num
      eric or --hexnumeric
    --nocolor: do not use colored output
    --can:               use the (ESD) CAN interface instead of RS232 serial port

    --net=NET            use the ESD CAN net number NET
    --id_read=ID_READ:   use CAN ID ID_READ for receiving CAN messages (default:
      43)
    --id_write=ID_WRITE: use CAN ID ID_WRITE for writing CAN messages (default: 4
      2)
    -j, --jtag:          use jtag_uart via nios2-terminal instead of RS232 serial
       port
    -i, --instance=ID:   use ID as jtag instance (see nios2-terminal --help)
    --cable=ID:          use ID as jtag cable (see nios2-terminal --help)
    --device=ID:         use ID as jtag device (see nios2-terminal --help)
```

8.3 Primary user interface classes

The primary user interface classes:

Classes

- class sdh.dsa.cDSA

  Interface class to access the DSACON32m, the tactile sensor controller of the SDH.

- class sdh.sdh.cSDH

  The actual SDH classes.

8.3.1 Detailed Description

The primary user interface classes:

Chapter 9

Namespace Documentation

9.1 Package demo-benchmark

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-benchmark.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2011 SCHUNK GmbH & Co. KG"
- int DEMO_BENCHMARK_USE_COMBINED_SET_GET = 1
- def CreateOptionParser

  Command line option handling:

- def GotoPose
- def Flat

  print flat representation of iterable l ([1,2,3] yields "1, 2, 3")

- def main

  The main function.

9.1.1 Function Documentation

9.1.1.1 def demo-benchmark.CreateOptionParser ( )

Command line option handling:

Create an option parser specifically for this demo program.

9.1.1.2 def demo-benchmark.Flat (*l*, *sep* = " ")

print flat representation of iterable l ([1,2,3] yields "1, 2, 3")

9.1.1.3 def demo-benchmark.GotoPose (*hand*, *ta*)

9.1.1.4 def demo-benchmark.main ( )

The main function.

Main function of demo script. Parses command line and reacts accordingly.

9.1.2 Variable Documentation

9.1.2.1 string demo-benchmark.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.1.2.2 string demo-benchmark.__copyright__ = "Copyright (c) 2011 SCHUNK GmbH & Co. KG"

9.1.2.3 string demo-benchmark.__doc__

Initial value:

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

- Example usage:
  - Start moving wildly using an SDH that is connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-benchmark.py --tcp=192.168.1.42:23


  - Start moving wildly using an SDH that is connected to:
    - port 2 = COM3 (joint controllers) and
    > demo-benchmark.py -p 2


  - Start moving wildly using an SDH that is connected to:
    - USB to RS232 converter 0 (joint controllers) and
    > demo-benchmark.py --sdh_rs_device=/dev/ttyUSB0


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

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

The docstring describing the purpose of the script:

9.1.2.4 string demo-benchmark.__url__ = "http://www.schunk.com"

9.1.2.5 string demo-benchmark.__version__ = "$Id: demo-benchmark.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.1.2.6 int demo-benchmark.DEMO_BENCHMARK_USE_COMBINED_SET_GET = 1

9.2 Package demo-calc-workspace

Functions

- def Print

Variables

- tuple parser

  Command line option handling:

- tuple types = dict(all=0, contour=1)
- string dest = "step0"
- string help = "Set step width for finger axis angle 0 to STEP, default=5."
- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)

  The actual script code:

- float phi = 90.0

Python specific variables

Some definitions that describe the script for python

Output a data file with xyz fingertip positions for all possible angles

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"

- string __version__ = "$Id: demo-calc-workspace.py 4355 2009-05-04 17:17:39Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.2.1 Function Documentation

9.2.1.1 def demo-calc-workspace.Print (*a0*, *a1*, *a2*)

9.2.2 Variable Documentation

9.2.2.1 string demo-calc-workspace.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.2.2.2 string demo-calc-workspace.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.2.2.3 string demo-calc-workspace.__url__ = "http://www.schunk.com"

9.2.2.4 string demo-calc-workspace.__version__ = "$Id: demo-calc-workspace.py 4355 2009-05-04 17:17:39Z Osswald2 $"

9.2.2.5 tuple demo-calc-workspace.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.2.2.6 string demo-calc-workspace::dest = "step0"

9.2.2.7 tuple demo-calc-workspace.hand = sdh.cSDH(options=options.__dict__)

The actual script code:

Create a global instance "hand" of the class cSDH according to the given options:

9.2.2.8 string demo-calc-workspace::help = "Set step width for finger axis angle 0 to STEP, default=5."

9.2.2.9 tuple demo-calc-workspace.parser

Initial value:

```
sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
                                 revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.2.2.10 float demo-calc-workspace.phi = 90.0

9.2.2.11 tuple demo-calc-workspace.types = dict(all=0, contour=1)

9.3 Package demo-contact-grasping

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-contact-grasping.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def CreateOptionParser

  Command line option handling:

- def GotoStartPose
- def main

  The main function.

9.3.1 Function Documentation

9.3.1.1 def demo-contact-grasping.CreateOptionParser ( )

Command line option handling:

Create an option parser specifically for this demo program.

9.3.1.2 def demo-contact-grasping.GotoStartPose (*hand*, *msg*)

9.3.1.3 def demo-contact-grasping.main ( )

The main function.

Main function of demo script. Parses command line and reacts accordingly.

9.3.2 Variable Documentation

9.3.2.1 string demo-contact-grasping.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.3.2.2 string demo-contact-grasping.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.3.2.3 string demo-contact-grasping.__doc__

Initial value:

```
'''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:
  - Start grasping using an SDH that is connected via Ethernet:
     The SDH and the tactile sensors have a common IP-Address,
     here 192.168.1.42. The joint controller is attached to the
     TCP port 23 and the tactile sensors to TCP port 13000 (default).
     (Requires at least SDH-firmware v0.0.3.2)
     > demo-contact-grasping.py --tcp=192.168.1.42:23 --dsa_tcp=


   - Start grasping using an SDH that is connected to:
     - port 2 = COM3 (joint controllers) and
     - port 3 = COM4 (tactile sensor controller)
     > demo-contact-grasping.py -p 2 --dsaport=3


   - Start grasping using an SDH that is connected to:
     - USB to RS232 converter 0 (joint controllers) and
     - USB to RS232 converter 1 (tactile sensor controller)
     > demo-contact-grasping.py --sdh_rs_device=/dev/ttyUSB0 --dsa_rs_device=/dev/ttyUSB1


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


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

The docstring describing the purpose of the script:

9.3.2.4 string demo-contact-grasping.__url__ = "http://www.schunk.com"

9.3.2.5 string demo-contact-grasping.__version__ = "$Id: demo-contact-grasping.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.4 Package demo-dsa

Classes

- class cMovingAverage

  Some additional classes and functions.

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-dsa.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def CreateOptionParser

  Create an option parser specifically for this demo program.

- def main

  The main function.

9.4.1 Function Documentation

9.4.1.1 def demo-dsa.CreateOptionParser ( )

Create an option parser specifically for this demo program.

9.4.1.2 def demo-dsa.main ( )

The main function.

Main function of demo script. Parses command line and reacts accordingly.

9.4.2 Variable Documentation

9.4.2.1 string demo-dsa.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.4.2.2 string demo-dsa.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.4.2.3 string demo-dsa.__doc__

The docstring describing the purpose of the script:

9.4.2.4 string demo-dsa.__url__ = "http://www.schunk.com"

9.4.2.5 string demo-dsa.__version__ = "$Id: demo-dsa.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.5 Package demo-GetAxisActualAngle

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-GetAxisActualAngle.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- string dest = "period"
- string help = "Time period of measurements in seconds. The default of '0' means: report once only."
- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)

  The actual script code:

- tuple t = hand.MoveHand(sequ=False)
- tuple start = sdh.time.time()
- tuple a_angles = hand.GetAxisActualAngle(sdh.All)
- tuple a_velocities = hand.GetAxisActualVelocity(sdh.All)
- tuple now = sdh.time.time()
- tuple xyz = hand.GetFingerXYZ(fi, None)

9.5.1 Variable Documentation

9.5.1.1 string demo-GetAxisActualAngle.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.5.1.2 string demo-GetAxisActualAngle.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.5.1.3 string demo-GetAxisActualAngle.__doc__

Initial value:

```
'''
Print current actual axis angles from SDH.

- Example usage:
  - Print actual angles of an SDH connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-GetAxisActualAngle.py --tcp=192.168.1.42:23

  - Print actual angles of an SDH connected to port 2 = COM3 once:
    > demo-GetAxisActualAngle.py -p 2

  - Print actual angles of an SDH connected to port 2 = COM3 every 500ms:
    > demo-GetAxisActualAngle.py -p 2 -t 0.5

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

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

The docstring describing the purpose of the script:

9.5.1.4 string demo-GetAxisActualAngle.__url__ = "http://www.schunk.com"

9.5.1.5 string demo-GetAxisActualAngle.__version__ = "$Id: demo-GetAxisActualAngle.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.5.1.6 tuple demo-GetAxisActualAngle.a_angles = hand.GetAxisActualAngle(sdh.All)

9.5.1.7 list demo-GetAxisActualAngle::a_velocities = hand.GetAxisActualVelocity(sdh.All)

9.5.1.8 tuple demo-GetAxisActualAngle.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.5.1.9 string demo-GetAxisActualAngle::dest = "period"

9.5.1.10 tuple demo-GetAxisActualAngle.hand = sdh.cSDH(options=options.__dict__)

The actual script code:

Create a global instance "hand" of the class cSDH according to the given options:

9.5.1.11 string demo-GetAxisActualAngle::help = "Time period of measurements in seconds. The default of '0' means: report once only."

9.5.1.12 tuple demo-GetAxisActualAngle.now = sdh.time.time()

9.5.1.13 tuple demo-GetAxisActualAngle.parser

Initial value:

```
sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
                                revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.5.1.14 tuple demo-GetAxisActualAngle.start = sdh.time.time()

9.5.1.15 tuple demo-GetAxisActualAngle.t = hand.MoveHand(sequ=False)

9.5.1.16 tuple demo-GetAxisActualAngle.xyz = hand.GetFingerXYZ(fi, None)

9.6 Package demo-gui

Classes

- class cTkSDHFinger

  A widget for a single finger.

- class cTkSDHSavePose

  A widget to save restore a single pose.

- class cTkSDHSavePoses

  A widget to save/restore all poses.

- class cTkSDHGrip

  A widget to access the grip skills stored in the SDH.

- class cTkSDHButtonBox

  A simple box for buttons.

- class cTkSDHMenu

  The Menu for the application.

- class cTkSDHPID

  Toplevel window to show and adjust the pid parameters of an SDH.

- class cTkSDHCurrent

  Toplevel window to show and adjust the motor current parameters of an SDH.

- class cTkSDHApplication

  The "Application" class of the simple SDH GUI.

- class cTkSDHInterfaceSelectorToplevel

  A toplevel widget class, used to select the communication interface to the SDH.

Python specific variables

Some definitions that describe the script for python

- string __doc__

The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-gui.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- we_have_can = True
- hand = None

  global variables

- dbg = None
- options = None
- root = None
- tuple persistent_settings = sdh.util.GetPersistantDict(name=".demo-gui-startsettings", cdbg = dbg)
- string schunk_logo
- def main

9.6.1 Function Documentation

9.6.1.1 def demo-gui.main ( )

9.6.2 Variable Documentation

9.6.2.1 string demo-gui.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.6.2.2 string demo-gui.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.6.2.3 string demo-gui.__doc__

Initial value:

```
'''
Simple GUI (Graphical User Interface) to command an SDH.

A window based interactive application with GUI (Graphical User
Interface) is started. It allows to command the SDH interactively
with mouse and keyboard. The application uses the sdh.py python import
library to connect to the SDH and the Tkinter package (Tk for python)
to build the gui elements.
If no RS232 port or CAN options to select the communication channel are
given on the command line then an interactive window to select the
channel is shown first before the actual program is started.

Apart from the intuitive gui elements the following keyboard shortcuts
can be used:
- <Pause/Break> : FastStop
- <CTRL-c>      : Close application after powering off the controllers
- <CTRL-C>      : Close application and keep controllers powered
- <CTRL-m>      : Move axes of hand to the values set with the sliders
```

```
- <CTRL-s>      : Stop movement of axes, but keep controllers enabled
- <CTRL-a>      : Set Sliders to the current actual angle of the corresponding
                  axis
- <CTRL-h>      : Set target position to "home" (all 0.0) Warning: collisions
                  are not checked!

- Example usage:
  - Start interactive GUI. Before the actual interface appears a
    window will appear to interactively select the interface to use.
    > demo-gui.py

  - Start interactive GUI using an SDH connected via Ethernet.
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-gui.py --tcp=192.168.1.42

  - Start interactive GUI using an SDH connected to port 2 = COM3.
    (The port for the tactile sensors can be choosen interactively)
    > demo-gui.py -p 2

  - 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-gui.py -p 2 --dsaport=3 -v
'''
```

The docstring describing the purpose of the script:

9.6.2.4 string demo-gui.__url__ = "http://www.schunk.com"

9.6.2.5 string demo-gui.__version__ = "$Id: demo-gui.py 12281 2014-09-30 07:44:33Z Osswald2 $"

9.6.2.6 demo-gui.dbg = None

9.6.2.7 demo-gui.hand = None

global variables

9.6.2.8 demo-gui.options = None

9.6.2.9 tuple demo-gui.persistent_settings = sdh.util.GetPersistantDict(name=".demo-gui-startsettings", cdbg = dbg)

9.6.2.10 demo-gui.root = None

9.6.2.11 string demo-gui.schunk_logo

Initial value:


```
"""R0lGODlhoQA6ANUAAEhhgMLK1IWWqho5YNHX3/Dy9KOwvwhKdAdYgzlUdQKT
wAZnkgSEsGd7lSpGagozXVhuigKbx+Dl6rK9yglCbAdgi3aJoJSitAhRewV2
oQOMuAo7ZAVumQR9qRpBZwGiz////wssVQAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACH5BAAAAAAALAAAAAChADoAAAb/
wJBwSCwaj8ikcslsOp/QqHRKrVqv2Kx2y+16v+CweEwum8/otHrNbrvf8Lh8
Tq/b7/YNft9+LCJ8gWd+ER8fgohiFYWGh4mPWwgKjY2Ql k4UB5oHD0KSlJSX
okYPFQyghgqTqJWjow8crLKsrqIYjLO5hrWWsbq/jryIGcDAwogVxcbHfBTK
vxliAABMDtPMRxrPsx16QgMWAhBZAAQg5yAEDUYNE+jnBRPrAe8gRQL1APTo
AUX75wEAvCuQwEg9ASHq9RsywNy7cSEQbGPFgFMFIQkIXAhn4EqDevyGZAT5
Tl+9e/n+gVg4RGXAegSLHEz4jmUIA/XWCTk1K8IC/wohHmDg2ahixEkYQkxw
IMHABHFVBhQgeQ7jVKrnTL5DWdKlv5oCYQ4gMlPhEHzvOgrZkEvDg3YBAlwY
IPGDgqSfDEXrJ+GCAQAIqVjAam+ABMJZVdojghadVoBf+YWtR2CskLI1hXzM
PKQuKwUPcA5MgABBiAPaKCkI0Y8ABHkWqjRON7bB1BCzzxEQAMCCgamPqzJO
ybklWKqVL79DaDYBZctCfLE6kBveAApEQb01sO6CBOhSZn//RiDE4XexiQy4
kEDxtPeiHXsl4nIyyOSY+Tm4Cq8gkeyqhcBfPRZ4hsoBIbw2wVJWQACTAJYN
YBYSiiEWnE2sHYdVZfnp5v+QY0YAWJR99UxwwCwIclGhBNSQGNgRFRJ2YWQA
kQiSAR+CwBxhL/4nCwM28nOiLClu0RBJDbiYRIxYzUifhujkSNKOWBUA3k6y
KCAhVQIYCEqRXAgwIAgFOJjWkogRh44ERQz4UknxTUlTlGNiGAIxFN1ElQMi
NuKNFtANECcIFozpHxEOOODeewAMGhwIh5L4JoiDLjdnjSD1GEIyoBhVwZE5
YdBTFwIwqBw6AgxagAAFDQABTo9y5digEkDggG0EkrgOqJleuhJuIFEzhDON
aIDgJxl48FuNB+CCSgek6objQUEGq5isWVVLkjUzhcCrpRMqZiURkyhgWl6p
mCb6BAV4zqLuFtW9U9AFacY6XElKpQnCBSEoSV6vE0o1oSemHdDnM4BES5Va
esp47b0gDsAkOmr5+y+4xdnYo8ETsXJRF+WQxC8REEh5jgQQPnxWPkM0cF49
EujUb7cMDUglZEPQy/IGHXTMigZhWCPA0BBcOYQDEAwtzqEJMCrs0U5fmUDS
Qzdw6DdO02CENe9p7fTV/X4dwgI+oxIBUNjgQXbZlCSVttpsGxLBu2/bsXbZ
CqBd9x13T+RTJ3vDPZEGFQAeuOCoGEUBBgs0XgEnhwvSd7qR19K3uZXzsnYE
H2du+d+eC0OB4aGXbvrpqKeu+uqst+7667UEAQA7"""
```


9.6.2.12 demo-gui.we_have_can = True

9.7 Package demo-radians

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-radians.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)
- tuple faa = hand.GetFingerActualAngle(0)

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple fta = list(faa)

9.7.1 Variable Documentation

9.7.1.1 string demo-radians.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.7.1.2 string demo-radians.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.7.1.3 string demo-radians.__doc__

Initial value:

```
'''
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10
using the "pose" (coordinated position) control mode.

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

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

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

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

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

The docstring describing the purpose of the script:

9.7.1.4 string demo-radians.__url__ = "http://www.schunk.com"

9.7.1.5 string demo-radians.__version__ = "$Id: demo-radians.py 11045 2013-11-27 15:12:49Z Osswald2 $"

9.7.1.6 tuple demo-radians.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.7.1.7 tuple demo-radians::faa = hand.GetFingerActualAngle(0)

do some prepartions: Switch to "pose" controller mode and set default velocities first:

9.7.1.8 tuple demo-radians::fta = list(faa)

9.7.1.9 tuple demo-radians.hand = sdh.cSDH(options=options.__dict__)

9.7.1.10 tuple demo-radians.parser

Initial value:

```
1 sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
2                                revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.8 Package demo-simple

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-simple.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)
- tuple faa = hand.GetFingerActualAngle(0)

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple fta = list(faa)

9.8.1 Variable Documentation

9.8.1.1 string demo-simple.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.8.1.2 string demo-simple.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.8.1.3 string demo-simple.__doc__

Initial value:

```
'''
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10
using the "pose" (coordinated position) control mode.

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

  - Make SDH connected to port 2 = COM3 move:
    > demo-simple.py -p 2

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

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

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

The docstring describing the purpose of the script:

9.8.1.4 string demo-simple.__url__ = "http://www.schunk.com"

9.8.1.5 string demo-simple.__version__ = "$Id: demo-simple.py 11045 2013-11-27 15:12:49Z Osswald2 $"

9.8.1.6 tuple demo-simple.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.8.1.7 tuple demo-simple::faa = hand.GetFingerActualAngle(0)

do some prepartions: Switch to "pose" controller mode and set default velocities first:

9.8.1.8 tuple demo-simple::fta = list(faa)

9.8.1.9 tuple demo-simple.hand = sdh.cSDH(options=options.__dict__)

9.8.1.10 tuple demo-simple.parser

Initial value:

```
1 sdh.cSDHOptionParser( usage    = __doc__ + "\nusage: %prog [options]",
2                                revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.9 Package demo-simple2

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-simple2.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)
- int iFinger = 0

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple faa = hand.GetFingerActualAngle(iFinger)
- tuple fta = list(faa)
- tuple t = hand.MoveFinger(iFinger, False)

9.9.1 Variable Documentation

9.9.1.1 string demo-simple2.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.9.1.2 string demo-simple2.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.9.1.3 string demo-simple2.__doc__

Initial value:

```
'''
Very simple python demo script using the sdh package.
Will move first finger 3 times between current position and current position-10.
The movement will be stopped in the middle of the movement.

- Example usage:
  - Make SDH connected connected via Ethernet move:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-simple2.py --tcp=192.168.1.42:23

  - Make SDH connected to port 2 = COM3 move:
    > demo-simple2.py -p 2

  - Make SDH connected to USB to RS232 converter 0 move:
    > demo-simple2.py --sdh_rs_device=/dev/ttyUSB0

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The joint controller is attached to the
    TCP port 23 and the tactile sensors to TCP port 13000.
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-simple2.py --tcp=192.168.1.42:23 --dsa_tcp=:13000 -v

  - Get the version info of both the joint controllers and the tactile
    sensor firmware from an SDH connected to:
    - port 2 = COM3 (joint controllers) and
    - port 3 = COM4 (tactile sensor controller)
    > demo-simple2.py --port=2 --dsaport=3 -v
'''
```

The docstring describing the purpose of the script:

9.9.1.4 string demo-simple2.__url__ = "http://www.schunk.com"

9.9.1.5 string demo-simple2.__version__ = "$Id: demo-simple2.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.9.1.6 tuple demo-simple2.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.9.1.7 tuple demo-simple2::faa = hand.GetFingerActualAngle(iFinger)

9.9.1.8 tuple demo-simple2::fta = list(faa)

9.9.1.9 tuple demo-simple2.hand = sdh.cSDH(options=options.__dict__)

9.9.1.10 int demo-simple2.iFinger = 0

do some prepartions: Switch to "pose" controller mode and set default velocities first:

9.9.1.11 tuple demo-simple2.parser

Initial value:

```
sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
                                revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.9.1.12 tuple demo-simple2.t = hand.MoveFinger(iFinger, False)

9.10 Package demo-simple3

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__
  The docstring describing the purpose of the script:
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-simple3.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

- tuple parser

  Command line option handling:

- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)

9.10.1 Variable Documentation

9.10.1.1 string demo-simple3.__author__ = ”Dirk Osswald: dirk.osswald@de.schunk.com”

9.10.1.2 string demo-simple3.__copyright__ = ”Copyright (c) 2007 SCHUNK GmbH & Co. KG”

9.10.1.3 string demo-simple3.__doc__

Initial value:

```
'''
Very simple python demo script using the sdh package.
Will move axes 1,2 and 3 to a fixed position, then
return back to the home position.
The movement will be started 'non sequentially' and
the scripts then waits for the movement to be finished.

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

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

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

   - Get the version info of both the joint controllers and the tactile
     sensor firmware from an SDH connected via Ethernet.
     The SDH and the tactile sensors have a common IP-Address,
     here 192.168.1.42. The joint controller is attached to the
     TCP port 23 and the tactile sensors to TCP port 13000.
     (Requires at least SDH-firmware v0.0.3.2)
     > demo-simple3.py --tcp=192.168.1.42:23 --dsa_tcp=:13000 -v

- Get the version info of both the joint controllers and the tactile
     sensor firmware from an SDH connected to:
     - port 2 = COM3 (joint controllers) and
     - port 3 = COM4 (tactile sensor controller)
     > demo-simple3.py --port=2 --dsaport=3 -v
'''
```

The docstring describing the purpose of the script:

9.10.1.4 string demo-simple3.__url__ = "http://www.schunk.com"

9.10.1.5 string demo-simple3.__version__ = "$Id: demo-simple3.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.10.1.6 tuple demo-simple3.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.10.1.7 tuple demo-simple3.hand = sdh.cSDH(options=options.__dict__)

9.10.1.8 tuple demo-simple3.parser

Initial value:

```
sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
                                  revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.11 Package demo-tactile

Classes

- class cTkSDHTactileApplication

  The "Application" class of demo-tactile.py, the simple SDH tactile visualizer.

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-tactile.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple _dbg = sdh.dbg.tDBG(True)
- def main

9.11.1 Function Documentation

9.11.1.1 def demo-tactile.main ( )

9.11.2 Variable Documentation

9.11.2.1 string demo-tactile.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.11.2.2 string demo-tactile.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.11.2.3 string demo-tactile.__doc__

Initial value:

```
'''
Simple GUI to visualize tactile sensors of SDH. (Python demo script using the sdh.py import library.)

- Example usage:
  - Display the tactile sensors connected via Ethernet.
    The SDH and the tactile sensors have a common IP-Address,
    here 192.168.1.42. The tactile sensors use TCP port 13000 (default).
    (Requires at least SDH-firmware v0.0.3.2)
    > demo-tactile.py --dsa_tcp=192.168.1.42

   - Display the tactile sensors connected to port 3 = COM4
     with the default style (colors, no numbers):
     > demo-tactile.py --dsaport=3

   - Display the tactile sensors connected to USB to RS232
     converter 0 with the default style (colors, no numbers):
     > demo-tactile.py --dsa_rs_device=/dev/ttyUSB0

   - Display the tactile sensors connected to port 3 = COM4
     with greycodes and numerical output in percent:
     > demo-tactile.py --dsaport=3 --style=grey --style=percent

   - 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-tactile.py -p 2 --dsaport=3 -v

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

The docstring describing the purpose of the script:

9.11.2.4 string demo-tactile.__url__ = "http://www.schunk.com"

9.11.2.5 string demo-tactile.__version__ = "$Id: demo-tactile.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.11.2.6 tuple demo-tactile._dbg = sdh.dbg.tDBG(True)

9.12 Package demo-temperature

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-temperature.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- string dest = "period"
- string help = "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 beginning of every line"
- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)

  The actual script code:

- tuple start = sdh.time.time()
- tuple L = hand.GetTemperature()

9.12.1 Variable Documentation

9.12.1.1 string demo-temperature.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.12.1.2 string demo-temperature.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.12.1.3 string demo-temperature.__doc__

Initial value:

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

- Example usage:
  - Print temperatures of an SDH connected via Ethernet:
    The SDH has IP-Address 192.168.1.42 and is attached to TCP port 23.
    (Requires at least SDH-firmware v0.0.3.1)
    > demo-temperature.py --tcp=192.168.1.42:23

  - Print temperatures of an SDH connected to port 2 = COM3 once:
    > demo-temperature.py -p 2

  - Print temperatures of an SDH connected to USB to RS232 converter 0 once:
    > demo-temperature.py --sdh_rs_device=/dev/ttyUSB0

  - Print temperatures of an SDH connected to port 2 = COM3 every 500ms:
    > demo-temperature.py -p 2 -t 0.5

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

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

The docstring describing the purpose of the script:

9.12.1.4 string demo-temperature.__url__ = "http://www.schunk.com"

9.12.1.5 string demo-temperature.__version__ = "$Id: demo-temperature.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.12.1.6 tuple demo-temperature.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.12.1.7 string demo-temperature.dest = "period"

9.12.1.8 tuple demo-temperature.hand = sdh.cSDH(options=options.__dict__)

The actual script code:

Create a global instance "hand" of the class cSDH according to the given options:

9.12.1.9 string demo-temperature.help = "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 beginning of every line"

9.12.1.10 tuple demo-temperature.L = hand.GetTemperature()

9.12.1.11 tuple demo-temperature.parser

Initial value:

```
1 sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
2                                 revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.12.1.12 tuple demo-temperature.start = sdh.time.time()

9.13 Package demo-velocity-acceleration

Variables

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-velocity-acceleration.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:

- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple hand = sdh.cSDH(options=options.__dict__)
- int axis_index = 2

  Preparations: Move the hand to a pose that is adequate for this demo:

- int velocity = 40
- position_reached = False

9.13.1 Variable Documentation

9.13.1.1 string demo-velocity-acceleration.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.13.1.2 string demo-velocity-acceleration.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.13.1.3 string demo-velocity-acceleration.__doc__

Initial value:

```
'''
Very simple python demo script using the sdh package.
Will move first finger in "velocity with acceleration ramp" control mode.

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

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

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

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

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

The docstring describing the purpose of the script:

9.13.1.4 string demo-velocity-acceleration.__url__ = "http://www.schunk.com"

9.13.1.5 string demo-velocity-acceleration.__version__ = "$Id: demo-velocity-acceleration.py 10351 2013-06-18 16:28:14Z Osswald2 $"

9.13.1.6 int demo-velocity-acceleration.axis_index = 2

Preparations: Move the hand to a pose that is adequate for this demo:

Do some movements with "velocity with acceleration ramp" controller type, move with different velocities and accelerations.

9.13.1.7 tuple demo-velocity-acceleration.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.13.1.8 tuple demo-velocity-acceleration.hand = sdh.cSDH(options=options.__dict__)

9.13.1.9 tuple demo-velocity-acceleration.parser

Initial value:

```
1 sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
2                                revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.13.1.10 tuple demo-velocity-acceleration::position_reached = False

9.13.1.11 int demo-velocity-acceleration.velocity = 40

9.14 Package demo-workspace

Variables

Python specific variables

Some definitions that describe the script for python

Move fingers to show workspace of SDH. (Python demo script using the sdh.py import library.)

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: demo-workspace.py 6269 2010-12-03 11:46:13Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple parser

  Command line option handling:
- tuple dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.
- tuple hand = sdh.cSDH(options=options.__dict__)

  The actual script code.
- tuple ata = map(hand.uc_angle.ToExternal, ata)
- tuple t = hand.MoveHand()

9.14.1 Variable Documentation

9.14.1.1 string demo-workspace.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.14.1.2 string demo-workspace.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.14.1.3 string demo-workspace.__url__ = "http://www.schunk.com"

9.14.1.4 string demo-workspace.__version__ = "$Id: demo-workspace.py 6269 2010-12-03 11:46:13Z Osswald2 $"

9.14.1.5 tuple demo-workspace.ata = map(hand.uc_angle.ToExternal, ata)

9.14.1.6 tuple demo-workspace.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

An object to print script-level debug messages, if requested.

9.14.1.7 tuple demo-workspace.hand = sdh.cSDH(options=options.__dict__)

The actual script code.

Create a global instance "hand" of the class cSDH according to the given options:

9.14.1.8 tuple demo-workspace.parser

Initial value:

```
sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options]",
                               revision = __version__ )
```

Command line option handling:

Create an option parser object to parse common command line options:

9.14.1.9 tuple demo-workspace.t = hand.MoveHand()

9.15 Package dsa

Python module to control the tactile sensors of the SDH (SCHUNK Dexterous Hand).

9.15.1 Detailed Description

Python module to control the tactile sensors of the SDH (SCHUNK Dexterous Hand).

This is a python import module. It is meant to be imported by other modules and scripts. It provides constants, functions and classes to communicate with the **DSACON32m**, the tactile sensor controller of a SDH (SCHUNK Dexterous Hand) device connected to a PC. The main user interface is provided via the class sdh.dsa.cDSA

9.15.2 Dependencies

The following standard python modules are used:

- struct, array, threading, time

The following non-standard python modules are used

- util, utils, dbg : common utilities, provided by SCHUNK
- serial : the pySerial module from `http://pyserial.sourceforge.net/`
- py.test : unit testing framework from `http://codespeak.net/py/current/doc/index.html`

9.15.3 Copyright

9.16 Package miniterm

Classes

- class cTkSDHInterfaceSelectorFrame

  A toplevel widget class, used to interactively select the communication interface of the miniterm.py app on start.

Functions

- def GetColor

  return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_-back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

- def GetPrompt
- def hex2

  Return the hexadecimal representation of an integer or long integer with 2 digits (hex2(5)->0x05)

- def cls

  Clear screen.

- def reader

  loop forever and copy serial->console

- def StringToInt

  return int of string s, e.g.

- def HexStringToInt
- def SendFromFile

  send data from file filename to serial port.

- def writer

  loop and copy console->serial until EOF character is found

- def usage
- def Exit

  if wait_for_key is False then just call exit.

- def main

Variables

- list d = os.environ["HOME"]
- tuple histfile = os.path.join(d, ".minitermhist")
- tuple prefix_keyboard = GetColor("blue")
- tuple suffix_keyboard = GetColor("normal")
- tuple prefix_serialin = GetColor("normal")
- tuple suffix_serialin = GetColor("normal")
- tuple prefix_message = GetColor("green")
- tuple suffix_message = GetColor("normal")
- tuple prefix_error = GetColor("red")
- tuple suffix_error = GetColor("normal")
- tuple prefix_warning = GetColor("magenta")
- tuple suffix_warning = GetColor("normal")
- string VT100_CLR_SCREEN = "\x1b[2J"
- string EXITCHARACTER = '\x04'
- int CONVERT_CRLF = 2
- int CONVERT_CR = 1
- int CONVERT_LF = 0
- int eModeAscii = 0
- int eModeNumeric = 1
- int eModeHexNumeric = 2
- mode = eModeAscii
- additional_ascii = False
- int numeric_length = 8
- prompt = None
- input_log_file = None
- inputfilename = None
- g_exiting = False

- g_reader_thread = None
- serialport = None
- string online_help
- convert_outgoing = CONVERT_CRLF

9.16.1 Function Documentation

9.16.1.1 def miniterm.cls ( )

Clear screen.

Needed in windows console since that does not understand VT100 commands

9.16.1.2 def miniterm.Exit (*val* = `1`, *wait_for_key* = `("win32" in sys.platform)`)

if wait_for_key is False then just call exit.

If True tehn loop until a key is pressed then exit. On cygwin/linux wait_for_key defaults to False, on windows it defaults to True. (This is usefull to be able to view error messages of python when called on windows)

9.16.1.3 def miniterm.GetColor (*c*)

return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

If the environment variable "SDH_NO_COLOR" is set then "" is returned. If the environment variable "OS" is WIN∗ or Win∗ and "OSTYPE" is not "cygwin" then "" is returned. (to prevent color output on windows consoles which cannot handle it). If the color is not found in the list of known colors then the string "" is returned.

9.16.1.4 def miniterm.GetPrompt ( )

9.16.1.5 def miniterm.hex2 (*n*)

Return the hexadecimal representation of an integer or long integer with 2 digits (hex2(5)->0x05)

9.16.1.6 def miniterm.HexStringToInt (*s*)

9.16.1.7 def miniterm.main ( )

9.16.1.8 def miniterm.reader ( )

loop forever and copy serial->console

9.16.1.9 def miniterm.SendFromFile (*filename*)

send data from file filename to serial port.

Data is sent as is with the following exceptions:

- an empty line (2 consecutive newlines) make the sending pause until the RETURN key is pressed

9.16.1.10 def miniterm.StringToInt (*s*, *base* = `10`)

return int of string s, e.g.

10 for "10" or "0xa"

9.16.1.11 def miniterm.usage ( )

9.16.1.12 def miniterm.writer ( )

loop and copy console->serial until EOF character is found

9.16.2 Variable Documentation

9.16.2.1 miniterm.additional_ascii = False

9.16.2.2 int miniterm.CONVERT_CR = 1

9.16.2.3 int miniterm.CONVERT_CRLF = 2

9.16.2.4 int miniterm.CONVERT_LF = 0

9.16.2.5 miniterm.convert_outgoing = CONVERT_CRLF

9.16.2.6 list miniterm::d = os.environ[”HOME”]

9.16.2.7 int miniterm.eModeAscii = 0

9.16.2.8 int miniterm.eModeHexNumeric = 2

9.16.2.9 int miniterm.eModeNumeric = 1

9.16.2.10 string miniterm.EXITCHARACTER = ’\x04’

9.16.2.11 miniterm.g_exiting = False

9.16.2.12 miniterm.g_reader_thread = None

9.16.2.13 tuple miniterm.histfile = os.path.join(d, ”.minitermhist”)

9.16.2.14 miniterm.input_log_file = None

9.16.2.15 miniterm.inputfilename = None

9.16.2.16 miniterm.mode = eModeAscii

9.16.2.17 int miniterm.numeric_length = 8

9.16.2.18 string miniterm.online_help

Initial value:

```
"""
miniterm onlinehelp:
F1 + RETURN: Show this help.
F2 + RETURN: Activate text mode:
             For "47 0x11" as input 8-9 bytes will be sent, the chars
             of the string plus linefeed chars.
F3 + RETURN: Activate numeric mode:
             For "47 0x11" as input 2 bytes will be sent, 42 and 17.
             For "256 65536" as input 5 bytes will be sent, 0,1, 0,0,1.
              I.E. little endian encoding with the fewest bytes necessary.
F4 + RETURN: Activate hex numeric mode
              For "47 11" as input 2 bytes will be sent, 71 and 17.
```

```
F5 + RETURN: Toggle additional ascii display in numeric mode
F6 + RETURN: Toggle prompt
F7 FILENAME + RETURN: save received data to file FILENAME
F7 + RETURN:          close a previously opened file
F8 FILENAME + RETURN: send data from file FILENAME
"""
```

9.16.2.19 tuple **miniterm.prefix_error** = GetColor(”red”)

9.16.2.20 tuple **miniterm.prefix_keyboard** = GetColor(”blue”)

9.16.2.21 tuple **miniterm.prefix_message** = GetColor(”green”)

9.16.2.22 tuple **miniterm.prefix_serialin** = GetColor(”normal”)

9.16.2.23 tuple **miniterm.prefix_warning** = GetColor(”magenta”)

9.16.2.24 **miniterm.prompt** = None

9.16.2.25 **miniterm.serialport** = None

9.16.2.26 tuple **miniterm.suffix_error** = GetColor(”normal”)

9.16.2.27 tuple **miniterm.suffix_keyboard** = GetColor(”normal”)

9.16.2.28 tuple **miniterm.suffix_message** = GetColor(”normal”)

9.16.2.29 tuple **miniterm.suffix_serialin** = GetColor(”normal”)

9.16.2.30 tuple **miniterm.suffix_warning** = GetColor(”normal”)

9.16.2.31 string **miniterm.VT100_CLR_SCREEN** = ”\x1b[2J”

9.17 Package postinstall_sdh

Functions

- def Log
- def Install

  define necessary functions

- def Remove

Variables

- do_debug = False

  simple logging mechanism for debugging the postinstall script.

- log = None
- args_ok = False

  "main" function that calls the functions from above according to command line

9.17.1 Function Documentation

9.17.1.1 def postinstall_sdh.Install ( )

define necessary functions

9.17.1.2 def postinstall_sdh.Log (*msg*)

9.17.1.3 def postinstall_sdh.Remove ( )

9.17.2 Variable Documentation

9.17.2.1 postinstall_sdh.args_ok = False

"main" function that calls the functions from above according to command line

9.17.2.2 postinstall_sdh.do_debug = False

simple logging mechanism for debugging the postinstall script.

(needed since this is called from the windows installer and cannot simply print to stdout)

9.17.2.3 tuple postinstall_sdh::log = None

9.18 Package sdh

Implementation of the python package to control a SDH (SCHUNK Dexterous Hand).

Packages

- package auxiliary
- package canserial
- package dbg
- package dsa
- package release
- package sdh
- package sdhbase

- package sdhserial
- package tcpserial
- package tkdsa
- package unit
- package util
- package utils

Variables

Python specific variables

Some definitions that describe the module for python

- string __doc__
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- __version__ = release.PROJECT_RELEASE
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.18.1 Detailed Description

Implementation of the python package to control a SDH (SCHUNK Dexterous Hand). This is a python package. It is meant to be imported by other modules and scripts. It provides constants, functions and classes to communicate with a SDH (SCHUNK Dexterous Hand) device connected to a PC. The sdh python import module is the interface for user applications to access the SDH.

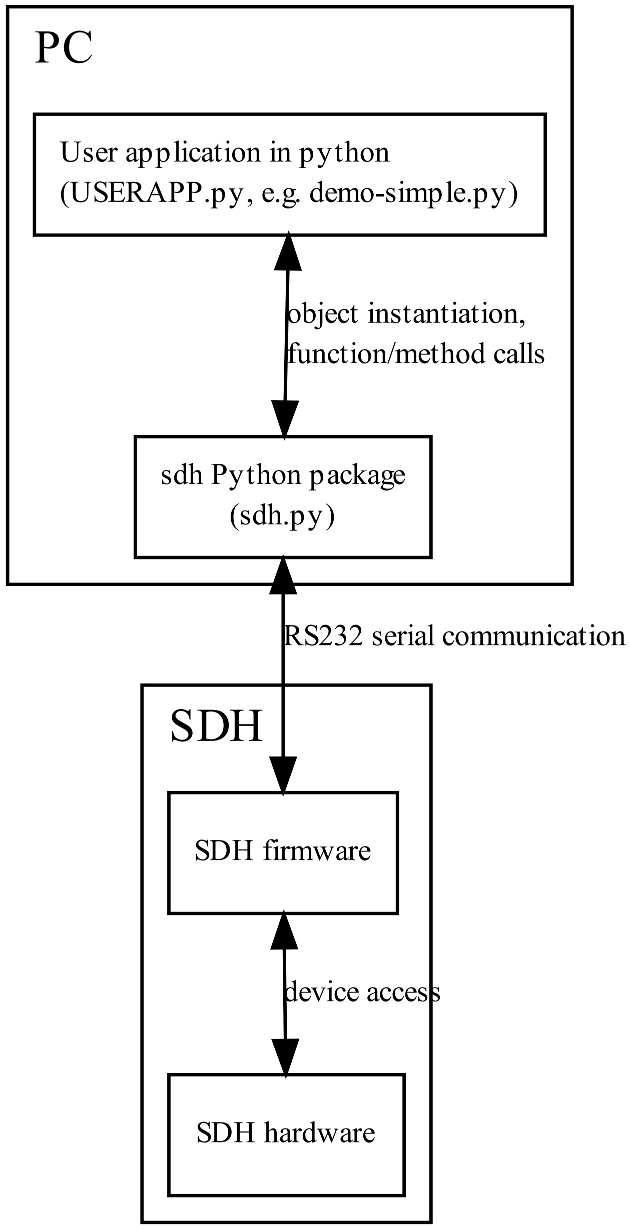
9.18.2 Overview

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 pyhton import library **"sdh.py"** that contains the complete sdh package and the python classes sdh.cSDH and **sdh.dsa.cDSA** with the main user interfaces. The sdh.sdh.cSDH class will be described in detail below.

Basic structure:

The basic structure of the components looks like this:


PC
User application in python
(USERAPP.py, e.g. demo-simple.py)
object instantiation,
function/method calls
sdh Python package
(sdh.py)
RS232 serial communication
SDH
SDH firmware
device access
SDH hardware

Basic architecture:

There are several classes defined here in sdh:

- sdh.cSDH is the primary 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 than that of other (low-level) classes.
- sdh.dsa.cDSA is the primary class to communicate with the tactile sensor controller DSACON32m within the SDH
- Other classes, like sdh.cSDHBase and sdh.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.
- sdh.cSDHError and derivatives: these are used when an exception is raised

Example use:

An exemplary use of the sdh package in a python script might look like this:

```
  ...
  # Import the sdh.py python import module:
  import sdh

  # Create an instance "hand" of the class cSDH:
  hand = sdh.cSDH()

  # Open communication to the SDH device via default serial port 0 == "COM1
"
  hand.Open()

  # Perform some action:
  #   get the current actual axis angles of finger 0
  faa = hand.GetFingerActualAngle( 0 )

  #   modify these by decreasing the proximal and the distal axis angles:
  fta = list(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 switches the axis controlle
rs off:
  hand.Close()
```

Real example code is available in the demo-∗.py demonstration scripts.

9.18.3 Dependencies

The sdh package makes use of many standard python packages like:

- sys, time, re, math, array, OptionParser
- Tkinter for demo-gui.py or demo-tactile.py

Additionally some 3rd party non-standard python packages are used. These are listed here.

9.18.4 Copyright

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

9.18.5 Variable Documentation

9.18.5.1 string sdh.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.18.5.2 string sdh.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.18.5.3 string sdh.__doc__

Initial value:

```
"""The python package to control a SDH (SCHUNK Dexterous Hand).

                The python __doc__ strings here provide only a very
                brief documentation, see the doxygen generated
                documentation (html or pdf) for details.

                In short: the cSDH class provides the end user interface to
                access a SDH. """
```

9.18.5.4 string sdh.__url__ = "http://www.schunk.com"

9.18.5.5 sdh.__version__ = release.PROJECT_RELEASE

9.19 Package sdh-ping

Python specific variables

Some definitions that describe the script for python

- string __doc__

  The docstring describing the purpose of the script:

- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: sdh-ping.py 6270 2010-12-03 11:49:03Z Osswald2 $"

- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def GetMedian
- def T2MS
- def avg
- def main

9.19.1 Function Documentation

9.19.1.1 def sdh-ping.avg (*iterable*)

Returns

average value of all values in *iterable*

9.19.1.2 def sdh-ping.GetMedian (*numericValues*)

Returns

median of iterable *numericValues*

9.19.1.3 def sdh-ping.main ( )

9.19.1.4 def sdh-ping.T2MS (*t*)

Returns

t (in seconds) converted to milliseconds

9.19.2 Variable Documentation

9.19.2.1 string sdh-ping.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.19.2.2 string sdh-ping.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.19.2.3 string sdh-ping.__doc__

Initial value:

```
'''
Measure response time of SDH

- Example usage:
  - Measure response time of SDH connected to port 2 = COM3 :
    > sdh-ping.py -p 2

  - Measure response time of SDH connected to ESD CAN 100 times:
    > sdh-ping.py --can  100

  - Measure response time of SDH connected via TCP 1000 times:
```

```
12     > sdh-ping.py --tcp=192.168.1.42 1000
13 '''
```

The docstring describing the purpose of the script:

9.19.2.4 string sdh-ping.__url__ = "http://www.schunk.com"

9.19.2.5 string sdh-ping.__version__ = "$Id: sdh-ping.py 6270 2010-12-03 11:49:03Z Osswald2 $"

9.20 Package sdh.auxiliary

Classes

- class cSDHOptionParser

  Customized OptionParser with some SDH specific options already set.

- class cSphere

  A class to represent sphere objects.

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "Auxiliary variables, functions, classes for sdh package"
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: auxiliary.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

Auxiliary variables, functions, classes

- has_dsa = True

  Auxiliary variables:

- float MIN_FLOAT = 3.40E+38
- float MAX_FLOAT = 3.40E+38
- def InIndex

  Auxiliary functions:

- def InRange

  Return True if v is in range [min_v .

- def ToRange

Return v limited to range [min_v .

- def Approx

  Return True if a is approximately the same as b.

- def InRange_a

  Return True if in list/tuple/array v=(v1,v2,...) each v_i is in range [min_v_i..max_v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

- def ToRange_a

  Return list/tuple/array v=(v1,v2,...) where each v_i is limited to range [min_v_-i..max_v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

- def Approx_a

  Return True if list/tuple/array a=(a1,a2,...) is approximately the same as b=(b1,b2,...).

- def DegToRad

  Return d in deg converted to rad.

- def RadToDeg

  Return r in rad converted to deg.

- def Square

  Return v squared (i.e.

- def Alltrue

  Return True if all elements of v (tuple, list, array.array).

- def Allmin

  Return list of min elements of v and w (tuple, list, array.array).

- def Allmax

  Return list of max elements of v and w (tuple, list, array.array).

- def AsStruct

  return dict_or_struct as struct

- def GetVersionInfo

  Return a string with all the version info:
  - *name and release of the calling script (PC),*
  - *name and release of the library (PC),*
  - *name of the platform (PC),*
  - *name and release of the Python executable (PC),*
  - *release and date of firmware (SDH),*
  - *release and date of SoC (SDH),*
  - *id and serial number of the SDH,*

  - *hardware and software versions and serial numbers for:*
    - *the tactile controller DSACON32m (SDH),*
    - *the tactile sensors (SDH)*

- def GetCommunicationInterfaceName
- def WriteIVFile

  Generate an OpenInventor iv file of all the objects by calling o.Toiv()

- def GetDevicePatterns

  Return a list of RS232 device name patterns corresponding to the current platform.

- def GetAvailablePorts

  Return a list of tuples (p,occupied), where p is device name of a serial port of the computer and occupied is True if the port is occupied by another application or the port is inaccessible.

- def GetIconPath

  Return a path to an appropriate icon for this application and OS.

- def PrettyStruct

  Return a string containing the name and the content of the structure s.

- def NumerifyRelease

  return a list of integer numbers for a release string

- def CompareReleases

  compare release strings rev1 and rev2.

9.20.1 Function Documentation

9.20.1.1 def sdh.auxiliary.Allmax (*v*, *w*)

Return list of max elements of v and w (tuple, list, array.array).

9.20.1.2 def sdh.auxiliary.Allmin (*v*, *w*)

Return list of min elements of v and w (tuple, list, array.array).

9.20.1.3 def sdh.auxiliary.Alltrue (*v*)

Return True if all elements of v (tuple, list, array.array).

9.20.1.4 def sdh.auxiliary.Approx (*a*, *b*, *eps*)

Return True if a is approximately the same as b.

I.E. |a-b| < eps

9.20.1.5 def sdh.auxiliary.Approx_a (*a*, *b*, *eps*)

Return True if list/tuple/array a=(a1,a2,...) is approximately the same as b=(b1,b2,...).

I.E. |a_i-b_i| < eps[i]

9.20.1.6 def sdh.auxiliary.AsStruct (*dict_or_struct*)

return dict_or_struct as struct

9.20.1.7 def sdh.auxiliary.CompareReleases (*rev1*, *rev2*)

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

9.20.1.8 def sdh.auxiliary.DegToRad (*d*)

Return d in deg converted to rad.

9.20.1.9 def sdh.auxiliary.GetAvailablePorts (*maxport* = `32`, *Print* = `lambda msg: None`, *device_patterns* = `GetDevicePatterns()`, *exclude* = `[]`)

Return a list of tuples (p,occupied), where p is device name of a serial port of the computer and occupied is True if the port is occupied by another application or the port is inaccessible.

The ports up to maxport are tested. If hints like "port is occupied" or "insufficient rights" should printed then Print should be a function that is able to print messages given as parameter. The device_patterns is a list of device name patterns like ["/dev/ttyS%d", "/dev/ttyUSB%d"] to search for.

Parameters

exclude	- a list of ports or devicenames that should be excluded from probing (required on Linux where searching for a port A that is already in use by the application makes a running select call on that port A block). The excluded ports will be contained as (name,True) in the return list.

Bug

SCHUNK-internal bugzilla ID: Bug 1013
On Linux the probing for available ports seems to disturb communication to already opened ports. This inhibits demo-gui.py from working.
=> Resolved in SDHLibrary-Python 0.0.2.2

9.20.1.10 def sdh.auxiliary.GetCommunicationInterfaceName (*options*, *dsa* = `False`)

Returns

the name of the selected communication interface according to *options* as a human readable string. If *dsa* is False then then interface for SDH is returned. If True then the interface for DSA is returned.

9.20.1.11 def sdh.auxiliary.GetDevicePatterns ( )

Return a list of RS232 device name patterns corresponding to the current platform.

9.20.1.12 def sdh.auxiliary.GetIconPath (*icon_dir* = `None`, *icon_base_name* = `"schunk"`)

Return a path to an appropriate icon for this application and OS.

Parameters

icon_dir	- a directory where the icon(s) are stored. The default None makes this function use the directory of the current application.
icon_base_name	- the base name of the icon without path prefix or file type suffix.

Returns

- For windows "icon_dir\icon_base_name.ico" will be returned if it exists.
- For linux "@icon_dir\icon_base_name.xbm" will be returned if it exists. If the file does not exist then None is returned. This way the result can be

given directly to wm_iconbitmap(sdh.GetIconPath()) from Tkinter.Tk() In recent (ca since spring 2012) tkinter on cygwin icons do not work any more. Background: X11 is needed (native windows tkinter does not work any more from python.

9.20.1.13 def sdh.auxiliary.GetVersionInfo (*script_name*, *script_release*, *options*, *hand* = None, *dsa_obj* = None)

Return a string with all the version info:

- name and release of the calling script (PC),
- name and release of the library (PC),
- name of the platform (PC),
- name and release of the Python executable (PC),
- release and date of firmware (SDH),
- release and date of SoC (SDH),
- id and serial number of the SDH,
- hardware and software versions and serial numbers for:
  - the tactile controller DSACON32m (SDH),
  - the tactile sensors (SDH)

9.20.1.14 def sdh.auxiliary.InIndex (*v*, *max_v*)

Auxiliary functions:

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

9.20.1.15 def sdh.auxiliary.InRange (*v*, *min_v*, *max_v*)

Return True if v is in range [min_v .

. max_v]

9.20.1.16 def sdh.auxiliary.InRange_a (*v*, *min_v*, *max_v*)

Return True if in list/tuple/array v=(v1,v2,...) each v_i is in range [min_v_i..max_v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

9.20.1.17 def sdh.auxiliary.NumerifyRelease (*rev*)

return a list of integer numbers for a release string

Parameters

rev	release string like "0.0.1.11-a"

Returns

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

9.20.1.18 def sdh.auxiliary.PrettyStruct (*name,* *s,* *exclude* = `None`)

Return a string containing the name and the content of the structure s.

s must be a struct or dictionary like object. For every element in s the key name and the value of the element is printed on one line, except for elements listed in list *exclude*.

9.20.1.19 def sdh.auxiliary.RadToDeg (*r*)

Return r in rad converted to deg.

9.20.1.20 def sdh.auxiliary.Square (*v*)

Return v squared (i.e.

v∗v)

9.20.1.21 def sdh.auxiliary.ToRange (*v,* *min_v,* *max_v*)

Return v limited to range [min_v .

. max_v]. I.e. if v is < min_v then min_v is returned, or if v > max_v then max_v is returned, else v is returned

9.20.1.22 def sdh.auxiliary.ToRange_a (*v,* *min_v,* *max_v*)

Return list/tuple/array v=(v1,v2,...) where each v_i is limited to range [min_v_i..max_-v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

I.e. if v_i is < min_v_i then min_v_i is part of the result list/tuple/array, or if v_i > max_v_i then max_v_i is part of the result list/tuple/array, else v_i is part of the result list/tuple/array

9.20.1.23 def sdh.auxiliary.WriteIVFile (*filename,* *objects*)

Generate an OpenInventor iv file of all the objects by calling o.Toiv()

9.20.2 Variable Documentation

9.20.2.1 string sdh::auxiliary.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.20.2.2 string sdh::auxiliary.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.20.2.3 string sdh::auxiliary.__doc__ = "Auxiliary variables, functions, classes for sdh package"

9.20.2.4 string sdh::auxiliary.__url__ = "http://www.schunk.com"

9.20.2.5 string sdh::auxiliary.__version__ = "$Id: auxiliary.py 12281 2014-09-30 07:44:33Z Osswald2 $"

9.20.2.6 sdh::auxiliary.has_dsa = True

Auxiliary variables:

Flag, if True then this hand has a DSA, i.e. a tactile sensor controller

9.20.2.7 float sdh::auxiliary.MAX_FLOAT = 3.40E+38

9.20.2.8 float sdh::auxiliary.MIN_FLOAT = 3.40E+38

9.21 Package sdh.canserial

Classes

- class tCANSerial

  Simple wrapper class to access an ESD CAN port like a serial port as a file like object.

9.22 Package sdh.dbg

Classes

- class tDBG

  A class to print debug messages.

9.23 Package sdh.dsa

Classes

- class cDSAError

  DSA (tactile sensor of the SDH) related error occurred.

- class cDSA

  Interface class to access the DSACON32m, the tactile sensor controller of the SDH.

Functions

- def LB

  return low byte of integer value i

- def HB

  return high byte of integer value i

- def Boolify

  return True if v != 0, else False

- def CRC16

  Do cyclic redundancy check calculation.

- def UIntFromBytes

  Return an int from the bytes in list the_bytes (1,2,3,4,...,bytes) in little endian.

- def FloatFromBytes

  Return a float from the list of the_bytes.

- def FloatToBytes

  Return a list of bytes from the float the_float.

- def UInt16ToBytes

  Return a list of bytes from the UInt16 the_uint16.

Variables

- All = None
- list gCRCtbl

  The CRC table used by the DSACON32m controller.

- int CRC_INIT_VALUE = 0xffff

9.23.1 Function Documentation

9.23.1.1 def sdh.dsa.Boolify (*v*)

return True if v != 0, else False

9.23.1.2 def sdh.dsa.CRC16 (*crc*, *byte*, *crc_table*)

Do cyclic redundancy check calculation.
Return the CRC for byte added to the current crc using the crc_table.

9.23.1.3 def sdh.dsa.FloatFromBytes (*the_bytes*)

Return a float from the list of the_bytes.

9.23.1.4 def sdh.dsa.FloatToBytes (*the_float*)

Return a list of bytes from the float *the_float*.

9.23.1.5 def sdh.dsa.HB (*i*)

return high byte of integer value i

9.23.1.6 def sdh.dsa.LB (*i*)

return low byte of integer value i

9.23.1.7 def sdh.dsa.UInt16ToBytes (*the_uint16*)

Return a list of bytes from the UInt16 *the_uint16*.

9.23.1.8 def sdh.dsa.UIntFromBytes (*the_bytes*)

Return an int from the bytes in list the_bytes (1,2,3,4,...,bytes) in little endian.

9.23.2 Variable Documentation

9.23.2.1 sdh::dsa.All = None

9.23.2.2 int sdh::dsa.CRC_INIT_VALUE = 0xffff

9.23.2.3 list sdh::dsa.gCRCtbl

Initial value:

```
[ 0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
                  0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
                  0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
                  0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
                  0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
                  0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
                  0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
                  0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
                  0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
                  0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
                  0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
                  0xdbfd, 0xcbdc, 0xfbbf, 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, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
                  0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
                  0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
                  0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
                  0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
                  0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
                  0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
                  0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
                  0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
                  0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
                  0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
                  0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
                  0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
                  0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
                  0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
                  0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
                  0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
```

]

The CRC table used by the DSACON32m controller.

9.24 Package sdh.release

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = """Definition and documentation of the project name and the release name ("version") for sdh package"""
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: release.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2013 SCHUNK GmbH & Co. KG"
- string PROJECT_NAME = "SDHLibrary-python"

  Define some variables.

- string FIRMWARE_RELEASE_RECOMMENDED = "0.0.3.3"
- string PROJECT_RELEASE = "0.0.2.9"

  Release name of the whole software project (a.k.a.

- string PROJECT_DATE = "2014-09-30"

  Date of the release of the software project.

9.24.1 Variable Documentation

9.24.1.1 string sdh::release.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.24.1.2 string sdh::release.__copyright__ = "Copyright (c) 2013 SCHUNK GmbH & Co. KG"

9.24.1.3 string sdh::release.__doc__ = """Definition and documentation of the project name and the release name ("version") for sdh package"""

9.24.1.4 string sdh::release.__url__ = "http://www.schunk.com"

9.24.1.5 string sdh::release.__version__ = "$Id: release.py 12281 2014-09-30 07:44:33Z Osswald2 $"

9.24.1.6 string sdh::release.FIRMWARE_RELEASE_RECOMMENDED = "0.0.3.3"

The recommended release of the firmware of an SDH used by this library

9.24.1.7 string sdh::release.PROJECT_DATE = "2014-09-30"

Date of the release of the software project.

The date of the release of the project.

9.24.1.8 string sdh::release.PROJECT_NAME = "SDHLibrary-python"

Define some variables.

Name of the software project.

The name of the "SDH" (SCHUNK Dextrous Hand) import-library for python.

9.24.1.9 string sdh::release.PROJECT_RELEASE = "0.0.2.9"

Release name of the whole software project (a.k.a.

as the *"version"* of the project).

The release name of the "SDHLibrary-python" project. The doxygen comment below contains the changelog of the project.

A suffix of "-dev" indicates a work in progress, i.e. a not yet finished release. A suffix of "-a", "-b", ... indicates a bugfix release.

From newest to oldest the releases have the following names and features:

- **0.0.2.9**: 2014-09-30
  - made library compatible with pyserial 2.7, which has yet another way of reporting errors
  - improved interface selection of demo-gui.py: now remembers last used setting; interface selector quit stops program as expected
  - added -nocolor option to miniterm.py
- **0.0.2.8**: 2014-02-28
  - added windows installer exe for Windows 64bit for SDHLibrary-python
  - now using --user-access-control=auto to create windows installers as bugfix for bug 1473: Bug: SDHLibrary-python Installer funktioniert nicht unter Windows 7 `https://192.168.101.101/mechatronik/show_bug.cgi?id=1473`
  - bugfix for bug 1517: Bug: assertion failure in util.cpp:207 in NumerifyRelease()
- **0.0.2.7**: 2013-11-27
  - fixed minor issues with miniterm.py
  - bugfix for bug 1462: Bug: limit checking is buggy when using radians `https://192.168.101.101/mechatronik/show_bug.cgi?id=1462`

        * fixed limit checking for angles, angular velcities and angular accelerations when using SDHLibrary with "Radians" as unit system, according to a bug report from Matei Ciocalie. Thanks.

- **0.0.2.6**: 2013-06-18
  - first version with recommended firmware 0.0.3.2 with ethernet TCP/IP support for tactile sensor data
- **0.0.2.5**: 2013-02-04
  - first version with recommended firmware 0.0.3.1 with ethernet TCP/IP support
- **0.0.2.4**:
  - bugfix corrected parameter calling order for ntcan.CIF.__init__()
  - made searching for COM ports work again in windows with new pyserial v2.6
  - made demo-gui.py/demo-tactile.py work in cygwin / tkinter (ignoring errors about invalid icons)
- **0.0.2.3**: 2012-02-18
  - enhanced miniterm.py to send data from file using command line parameters
  - made tactile sensor calibration stuff work again in demo-dsa.py
  - Enhancement `Bug 1088: Task: Implement functions to check compatibility of SDH firmware`
    * added cSDH.CheckFirmwareRelease() and cSDH.GetFirmwareReleaseRecommended() to be able to check the actual versus the recommended SDH firmware release. Needed for compatibility with the C++ version where it was added for the new SDH driver for ROS, see `http://www.ros.org/doc/api/cob_sdh/html/index.html`
- **0.0.2.2**:
  - fixed `Bug 1013: Bug: python script demo-gui.py does not show window on Linux`
    * probing for serial ports on linux disturbs communication to already opened ports
  - fixed serial.readline issue: On newer Linuxen the readline function available does no longer accept the keyword parameter eol. Removed since the default "\n" was used anyway
  - fixed readline module issue: made use of readline module optional in miniterm.py. On some Linuxen that module might not be available.
  - fixed virtual port issue: the "virtual" port -1 could not be specified on the command line

- **0.0.2.1**:
  - added `Bug 996: Task: Version numbering of DSACON32m firmware has changed since 2011-02-15` software version numbers for DSACON32m are reported correctly for the new firmwares
  - fixed `Bug 983: Bug: demo-dsa.py does not work on some laptops` The default baudrate for the RS232 port was not set correctly. So if the port was configured correctly before everything was OK, but if the port used another baudrate before then no communication was possible.
  - fixed `Bug 703: Bug: Tactile sensor frames cannot be read reliably in single frame mode`
    - Some firmware versions of the DSACON32m are not able to do single frame acquisition and enter push-mode. This might fill up the RS232 input buffer and leads to problems like reading of outdated frames or frame loss.
    - Added workaround to stop push mode immediately after it was entered unintentionally
- **0.0.2.0**: 2011-02-08
  - added support for communication via TCP (requires at least SDH firmware 0.0.3.0)
    - enhancement `Bug 874: Task: Enable TCP communication in SDHLibrary`
  - added support for special "-2fo" and "-dev" firmware version `Bug 915: Bug: SDHLibrary-python does not work with 2fo version of firmware`
  - renamed cSDH.OpenRS232() to cSDH.Open() since the open function can now handle CAN and TCP besides RS232. Kept OpenRS232() for compatibility reasons as well
  - Doxygen documentation is now generated with doxygen-1.7.3 with included javascript search engine
- **0.0.1.21**: 2010-05-11
  - enhanced demo-gui.py: now the motor current limits can be adjusted
  - renamed EmergencyStop stuff to FastStop (according to new SMP nomenclature)
  - Enhancement made acquiring of single tactile sensor frames available
- **0.0.1.20**: 2010-04-12
  - bugfix: `Bug 680: cDSA fails to communicate with Release 276 of DSACON32m Firmware`
- **0.0.1.19**: 2010-03-05

  - changed command line parameters for demo-dsa.py, demo-tactile.py, demo-gui.py regarding the tactile sensor adjustment for sensitivity and threshold. See online help for details.
    * Now a specific sensor can be modified independently.
    * Now the sensor parameters can be reset to the factory default.

- **0.0.1.18**: 2010-02-02
  - Added setting/getting of controller PID parameters from demo-gui.py (see Debug->PID adjust)
  - bugfix (firmware 0.0.2.10): `Bug 630: Bug: setting of pid parameters does not work`

- **0.0.1.17**: 2009-11-07
  - added online help of demonstration programs to doxygen documentation
  - added description of DSACON32m update process to SDH2_configuration-and-update.doc

- **0.0.1.16**: 2009-10-05
  - 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
  - bugfix: `Bug 389: miniterm.py --can states "Operation timed out" when called from windows python`
  - bugfix: `Bug 452: current demo-gui.py fails when connected to an old SDH v0.0.2.0` added firmware version specific code to sdh.cSDH.GetAxisLimitAcceleration()
  - enhancement: `Bug 456: Add support for adjust sensitivity of tactile sensors"`
    * added cDSA.SetMatrixSensitivity()
    * added command line options to demo-dsa.py to adjust matrix sensitivity
    * added support for new adjust threshold of tactgile sensors as well (needs DSACON32m Firmware >= R268)
    * the sensitivity and the threshold can now be saved persistently to configuration memory of the DSACON32m
    * added appropriate command line options to demo-dsa.py, demo-tactile.py and demo-gui.py
  - enhancement <a href="`https://192.168.101.101/mechatronik/show_bug.cgi?id=479`>Bug 479: Add support for new --sdh_rs_device and --dsa_rs_device command line options in SDHLibrary python
    * The actual device names are now stored in auxiliary.cSDHOptionParser.port and auxiliary.cSDHOptionParser.dsaport after command line option parsing

        * Changed the routines to check for available ports as well
        * On native windows available ports are listed as COMx (while command line option -p 0 still means COM1, even on native windows)
        * On cygwin available ports are listed as /dev/ttySx
        * On linux available ports are listed as /dev/ttySx and /dev/ttyUSBx
    - bugfix: `Bug 481: demo-gui handling of guidat files is unintuitive`
        * Now the guidat extension is automatically added if not explicitly given on save
        * the user can select "all (*.*)" on load
    - 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

- **0.0.1.15**: 2009-06-17
    - bugfix: `Bug 410: demo-gui.py --can fails`
    - bugfix: demo-gui.py "NUMBER_OF_GRIPS" is not a valid grip
    - bugfix: demo-gui.py did not work when hand was not in eCT_POSE controller type
    - internal: bugfix: made py.test work again
    - internal: enhancement: got rid of cygwin symbolic links which confused eclipse and tortoise
    - internal: moved testing stuff to test/subdir
    - 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
    - corrected doxygen description of GetTemperature()
    - bugfix: `Bug 433: Invalid negative velocities remain set when switching from speed based controllers back to pose controller` Adjusted documentation for SetController() accordingly
    - internal enhancement: added --parameter option to sdhrecord.py to specify which parameters to record
    - internal enhancement: made jtagserial.py use windows paths when called from a native windows interpreter. But miniterm.py -j still does not work.

- bugfix `Bug 411: Calling "perform grip" from demo-gui.py produces tracebacks when connected via CAN`
  - problem was actually in canserial where the timeouts where not handled properly
  - fixed that and enhanced documentation of tCANSerial
- enhancement: `Bug 439: enhance demo-contact-grasping`
  - now using the new velocity with acceleration ramp controller for this demo which makes it run much smoother
  - grasping with 2 fingers only, middle finger can be used as safety switch to end the demo
- bugfix: error replies from the firmware were ignored or reported as cSDHErrorCommunication, which triggered a resend
  - this is not adequate e.g. for range errors, so now for such errors a cSDHErrorInvalidArgument is raised which is not handled on library level (but can and should be handled on application level)
- internal: added test_v_limit to check bug 440
- enhancement: `Bug 445: demo-gui.py generates sdh.iv whereever its called from`
  - added parameter --ivfile to demo-gui.py to keep demo-gui.py from generating the (very limited usefull) sdh.iv file automatically
- enhancement: `Bug 442: acceleration limits cannot be queried from the firmware`
  - added new commands to read acceleration limits from firmware
- internal: added test_con to check con() and bug 433, added test_alim, added test_GripHand
- internal: added --interactive to pytest_options to be able to do py.test tests step by step
- 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

- **0.0.1.14**: 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.py
    - the allowed lower limits for velocity and acceleration now have to be adjusted when the controller type changes.

- enhancement: a cSDH object now has a release_firmware string member after connection to an SDH. This can be used together with new auxiliary.CompareReleases() member function for version specific code.
- bugfix: `Bug 404: reactivate grip hand stuff in demo-gui.py` since selgrip and grip work again in firmware 0.0.2.6

• **0.0.1.13**: 2009-05-05

- bugfix: `Bug 372: Disabled colored debug output on windows consoles` for better readability of debug messages on native windows
- bugfix: `Bug 377: demo-dsa.py -c / -s / -m do not work properly`
- replaced "== None" / "!= None" with "is None" / "is not None" according to PEP 8 see `http://www.python.org/dev/peps/pep-0008/`
- prefixed non public helper functions in cDSA with '_' to mark them as internal
- many minor code changes according to pylint recommendations and conventions
- now using doxypy filter in order to be able to use native python docstrings for documentation, see `http://code.foosel.org/doxypy`
- added Doxyfile to distribution
- enhancement: the "-v" command line option now tries to read and print the version info of the tactile sensors as well.
- the generated documentation has been improved by better grouping, cross linking and generally more documenting.
- changed internal package imports to use relative import command according to PEP 328: `http://docs.python.org/whatsnew/2.5.html#pep-328-absolut`
- bugfix: `Bug 380: Graphical display of tactile sensors does not work in demo-gui.py` Now works, see also demo-tactile.py for a standalone tactile sensor graphical display demo
- bugfix: `Bug 304: demo-dsa.py does not work at all` Working again, will work more robust
- bugfix: `Bug 338: demo-contact-grasping does not work` GetContactForce and GetContactArea do work now. Grasping does work now, althoug still not perfect. As long as only position controller is available in the SDH firmware the movements will be somewhat 'jerkily'.
- bugfix: `Bug 387: index file for sdhlibrary-python doxygen documentation installed with windows installer is wrong`
- bugfix: `Bug 388: miniterm.py installed with windows installer does not find CAN`
- bugfix: `Bug 376: First call to demo-dsa after powering SDH fails for python also`

• **0.0.1.12**: 2009-04-17

  - Bugfix: `Bug 335: importing of module canserial is wrong`

- **0.0.1.11**:
  - removed non working skill selection stuff from demo-gui.py
  - added interface selection window to demo-gui.py You can now start the demo-gui and miniterm.py without -p parameter and you will be queried for the RS232 port to use
  - added schunk icon to demo-gui.py
  - changed sdh.auxiliary.GetAvailablePorts() now it returns a list of tuples of (portnumber, occupied) instead of just the portnumber. needed for new demo-gui.py interface selection window
  - windemo dir is no longer included in the distribution (not usefull any longer)
  - miniterm.py now also has the possibility to ask interactively for the communication channel to use
  - updated copying of misc packages into distribution

- **0.0.1.10**:
  - bugfix: `Bug 322: AxisTargetVelocity cannot be set higher than 100 deg/s`
  - enhancement: display of ° (degree symbol) changed to use unicode strings, so that it displays correctly when used with native windows python interpreter
  - demo-gui.py sets the velocity according to the velocity slider, but reduces the velocity axis-wise if the slider exceeds a specific axis' velocity
  - enhancement: `Enhancement 315: Add documentation files to distribution`

- **0.0.1.9**:
  - bugfix: `Bug 114: demo-gui erlaubt Bewegungen, obwohl Kollisionserkennung "rot" anzeigt`
  - enhancement: demo-gui.py now remembers the last directory used when saving/loading *.guidat pose files
  - enhancement: < href="https://192.168.101.101/mechatronik/show_bug.cgi?id=269"> Bug 269: Add shortcut to uninstaller of python windows package to start menue

- **0.0.1.8**: 2008-10-14
  - added cSDHSerial.vlim() and cSDH.GetAxisLimitVelocity() to read velocity limits
  - corrected / enhanced setup.py:
    - docu is now included as pdf not doc
    - guidat file is include in distro

    * SCHUNK logo in windows installer
  - corrected / enhanced postinstall_sdh.py
    * bugfix: Bug: SDHLibrary/sdhflash generated shortcuts in py windows installers do not work if scripts are in dirs with spaces
  - included html footer in doc into SVN
  - enhancement: Enable debugging to logfile in SDHLibrary-python and in all demo scripts
  - added auxiliary.GetAvailablePorts() to scan for available RS232 ports
  - enhanced the printing of SDHLibrary and SDH firmware version info. besides release numbers the release dates, python interpreter info and SoC info is printed
  - demo-gui.py enhancements:
    * SDH version info can be displayed in a popup
    * only really available ports are listed in the menus, but still changing of port at runtime is not possible
    * disabled menus for not longer needed ref commands
  - made jtagserial.py more robust. Added auto-restart in case of error, e.g. due to SDH power cycle this way you don't have to restart the error log via jtag every time you restart the SDH
  - added missing cSDH.IsOpen()
  - enhancement: added long missing command line option support for python/demo/demo-simple* scripts

- **0.0.1.7**: 2008-08-08
  - while chasing `Bug 125: Small joint velocities cause movements to be stopped before the target position is reached`
    * added more tests to Makefile-plot
    * corrected ramp.py
  - added jtagserial to enable miniterm.py to use the jtag-uart
  - added support for new firmware v0.0.2.0 commands
    * soc, soc_date, ver_date
    * GetDuration
  - corrected / updated eErrorCode pseudo enum in cSDHBase
  - corrected cases where parameter All was not handled correctly in axis related commands in sdhserial
  - made py.test work again and added a few more tests

- **0.0.1.6**: 2008-06-13
  - minor changes to keep python DSA interface compatible to new C++ DSA interface

- **0.0.1.5**: 2008-06-06

  - made the reading of tactile sensor info work again (with DSACON32m Firmware 143)

- **0.0.1.4**: 2008-05-16
  - added support for new temperature sensors in library and demos

- **0.0.1.3**: 2008-03-19
  - enhanced windows installer: creates shortcuts to demo-gui and docu

- **0.0.1.2**: 2008-02-29
  - release for preliminary support of SDH2
    - CAN based communication on Windows with ESD card

- **0.0.1.1**: 2007-12-27
  - release for RoboCluster, Denmark
    - fixes to make library work with SDH-003

- **0.0.1.0**: 2007-08-30
  - restructured sdh package completely
    - The 5600 lines of the monolithic sdh.py were broken into more manageable units
    - made it a real python package with __init__.py and stuff
    - package contents were moved from ./sdh.py to ./sdh/*
    - demo scripts were moved to ./demo/
    - data files were moved to ./data/
  - Made makefiles more modular:
    - extracted generation of distribution to Makefile-dist
    - not all sub-makefiles need to be distributed any more (esp. Makefile-doc and Makefile-dist can be kept secret)
  - added support for new firmware features (for IBMT):
    - referencing of axes against mechanical block
    - saving of axis positions to non volatile memory

- **0.0.0.12**: 2007-06-11
  - added support for new firmware features: getting and setting of min/max/offset angles

- **0.0.0.11-a**: 2007-06-06
  - Release modified according to bug report from Martin Huelse (Uni Wales)
    - changes were needed almost exclusively in the C++ part

- Release including bugfixes for

0.0.0.11: 2007-05-24

- Release for care-o-bot (IPA, Stuttgart), Mai 2007
- Added missing MoveAxis() command
- Added max_angular_acceleration_a
- Added dsa.py: module for accessing DSA (Tactile Sensor Controller)
  - Added command line options for dealing with DSA (Tactile Sensor Controller) (adjusted options of other scripts to make them consistent)
  - Added demo scripts for tactile sensor access demo-dsa.py, demo-tactile.py, demo-contact-grasping.py
- Added setup.py to create a distribution with distutils
- while preparing release for IPA care-o-bot:
  - since line endings are corrected in firmware now removed the special EOL treatmnt in readline
  - enhanced generation of distribution
  - extended/added README files
  - added demo-simple3 in cpp and python
  - added missing demo-simple2.py
  - 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
  - made demo-GetAxisActualAngle.py work again when -s option is not given
  - removed use of Numeric.array in sdh.py , now using array.arrray since Numeric package is not available everywhere (test_sdh.py still uses Numeric)
  - tried to find bug:
    - firmware not moving from 5,-5,0,0,0,0,0 to 20,0,0,0,0,0,0:
    - axis 1 is stuck at 1.4...
    - bug could not be resolved (does not happen for for larger movements)
- **0.0.0.10**: 2007-04-05 -Release for demo at ICRA2007 (Rome), April 2007
  - Added support for the new commands of the firmware:
    - "A" command to get/set acceleration: Get/SetAxisTargetAcceleration()
    - "VP" command to get/set velocity profile: Get/SetVelocityProfile() and eVelocityProfile enums
    - "VEL" command to get actual velocity: GetAxisActualVelocity

    * new unit converter for angular accelerations
  - Added internal collision detection from Pedro Glogowski
    * CheckFingerCollisions() and helper functions
    * added check_collisions parameter to MoveFinger() and MoveHand()
    * incorporated calculating_angles.py into sdh.py
    * added helper functions like Square(), _AnglesToRad()
    * helper class cSphere
    * new exception cSDHErrorInternalCollision
    * added parameters to _GetFingerXYZ()
  - enhanced demo-gui.py:
    * looping over selected poses is possible
    * correct file extension is set in load/save file dialogs
    * SetToActual can now be made cyclic
    * Setting of Velocity profile is included but yet disabled since still buggy in the firmware

- **0.0.0.9**: 2007-03-19
  - Release for demo at NASA, march 2007
    * enhanced Makefiles slightly (better dist generation)
    * adjusted expected lines for "m" command (it now prints one line debug output for every axis)
    * added -m/--move and -V/--velocity command line options to demo-GetAxisActualAngle.py now this script can be used to record movements
    * added script demo-torquemeasurement for measuring axes torques
    * added word doc Startup.doc (forgotten from previous release 0.0.0.7 for Uni Wales)
    * corrected syntax error in sdh.py (introduced at visit Uni Wales)

- **0.0.0.8**: 2007-03-09
  - Release modified at visit Uni-Wales, march 2007
    * Changes to make everything work on Ubuntu-Linux
    * Enhanced Makefile a little bit to be more comfortable for the end user

- **0.0.0.7**: 2007-03-07
  - Release for Uni-Wales, march 2007

- **0.0.0.6**:
  - final release for SDH firmware 0.1

- **0.0.0.5**: 2007-02-12

  - enhanced doxygen docu for functions (added hint for unit conversion, where appropriate)
  - added/updated gnuplot scripts / makefile for visualizing the workspace
  - For RoboCluster on 2007-02-08:
    * added dist target to Makefile
    * added Inbetriebnahme.doc docu
    * added eye-candy *.bat and icon files
  - starting with kinematics calculation
    * manually deduced and implemented forward kinematics
    * finger angles -> fingertip position calculation in sdh.py
    * display functions for cartesian workspace in new demo-calc-workspace.py
    * new target in Makefile-plot to visualize the workspace with gnuplot

- **0.0.0.4**: 2007-02-06
  - sdh.py
    * Set/GetAxisMotorCurrent() for all modes (move, grip, hold)
    * added unit converter for motor current
    * added GripHand()
  - demo-gui.py:
    * temperature display
    * poses are converted to internal unit when saved and converted back to external when loaded
    * added menu to change ports/unit-systems/debug-settings interactively
    * now using cSDH only (no more cSDH.interface cheating)

- **0.0.0.3**: 2007-02-05
  - demo-gui.py:
    * SCHUNK logo
    * Grip/selgrip widget
    * Save/restore Pose widget
  - sdh.py:
    * external:
      · in cSDH now all fingers have NUMBER_OF_AXES_PER_FINGER=3 axes (finger 1 has a virtual axis), this simplifies the use very much, like e.g. in demo-gui.py
      · added extra arrays to cSDH that include min/max angles/velocities for the virtual axis
      · added Set/GetAxisMotorCurrent() - yet untested
      · added GetAxisMin/MaxAngle and tests
      · added GetFingerMin/MaxAngle and tests
      · added GetGripMaxVelocity - yet untested
    * internal:

· renamed cSDHErrorInvalidArgument to cSDHErrorInvalidParameter (more intuitive)
· changed cSDHBase.eErrorCode from lightweight object (utils.Struct) to normal dictionary to ease iteration over elements
· changed cSDHBase.eGraspId from lightweight object (utils.Struct) to normal dictionary to ease iteration over elements
· added CheckRange() CheckIndex() to check parameter and raise cSDHErrorInvalidParameter
· replaced asserts by CheckRange()/ CheckIndex() so that a cSDHErrorInvalidParameter is raised in case of error, this way the error messages can be much more descriptive and hintfull
· _ToIndexList now checks the indices and raises exception if invalid
· cSDHSerial now uses a "virtual" port if options["port"] < 0 (for testing offline without a SDH connected)
· added doxygen comment describing the axis indices and finger axis indices

- added demo-endurance-run.py

- **0.0.0.2**: 2007-01-31
  - many many changes while extending and refactoring sdh.h
    * now using cEnhancedSerial instead of hackish myreadline() workaround
    * Added customized OptionParser cSDHOptionParser to simplify handling of the usual command line parameters in (demo) scripts
    * Changed handling of passing options to cSDHBase and derived classes: options can now be given as a dictionary of keyword:value pairs. This way the output from the cSDHOptionParser can be used directly which simplifies and unifies (demo) scripts very much.
    * Added cUnitConverter class and predefined unit converter objects for setting/getting parameters in user or application specific unit sytems
  - Enhanced doxygen documentation very much
    * dot graph for overview
    * many code examples
- **0.0.0.1**: 2007-01-19
  - Initial "release" of the code
    * Doxygen documentation can be generated
    * Some functionality of sdh.py already available
    * all tests defined in test_sdh.py are OK (everything green)

9.25 Package sdh.sdh

Classes

- class cSDH

The actual SDH classes.

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "python module with end user interface to control a SDH (SCHUNK Dexterous Hand)"
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: sdh.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.25.1 Variable Documentation

9.25.1.1 string sdh::sdh.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.25.1.2 string sdh::sdh.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.25.1.3 string sdh::sdh.__doc__ = "python module with end user interface to control a SDH (SCHUNK Dexterous Hand)"

9.25.1.4 string sdh::sdh.__url__ = "http://www.schunk.com"

9.25.1.5 string sdh::sdh.__version__ = "$Id: sdh.py 11045 2013-11-27 15:12:49Z Osswald2 $"

9.26 Package sdh.sdhbase

Classes

- class cSDHError

  Exception classes.

- class cSDHErrorCommunication

  SDH-exception: Communication error occured in the sd module.

- class cSDHErrorInvalidParameter

  SDH-exception: Invalid parameter(s) were given.

- class cSDHErrorTimeout

  SDH-exception: A (communication) timeout occured.

- class cSDHErrorInternalCollision

  SDH-exception: The given target angles would lead to an internal collision.

- class cSDHBase

  The base class to control the SCHUNK Dexterous Hand.

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "Base classes for sdh package"
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: sdhbase.py 6432 2011-02-08 13:53:00Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- All = None

  A constant to address all fingers/axes when used as a parameter in certain functions.

9.26.1 Variable Documentation

9.26.1.1 string sdh::sdhbase.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.26.1.2 string sdh::sdhbase.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.26.1.3 string sdh::sdhbase.__doc__ = "Base classes for sdh package"

9.26.1.4 string sdh::sdhbase.__url__ = "http://www.schunk.com"

9.26.1.5 string sdh::sdhbase.__version__ = "$Id: sdhbase.py 6432 2011-02-08 13:53:00Z Osswald2 $"

9.26.1.6 sdh::sdhbase.All = None

A constant to address all fingers/axes when used as a parameter in certain functions.

9.27 Package sdh.sdhserial

Classes

- class cSDHSerial

  The class to communicate with a SDH via RS232.

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "Class to access SDH via RS232"
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: sdhserial.py 11438 2014-02-28 14:24:55Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.27.1 Variable Documentation

9.27.1.1 string sdh::sdhserial.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.27.1.2 string sdh::sdhserial.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.27.1.3 string sdh::sdhserial.__doc__ = "Class to access SDH via RS232"

9.27.1.4 string sdh::sdhserial.__url__ = "http://www.schunk.com"

9.27.1.5 string sdh::sdhserial.__version__ = "$Id: sdhserial.py 11438 2014-02-28 14:24:55Z Osswald2 $"

9.28 Package sdh.tcpserial

Classes

- class tTCPSerial

  Simple wrapper class to access a TCP port like a serial port as a file like object.

9.29 Package sdh.tkdsa

Classes

- class cSDHTactileSensorPatch

  A class to store a tactile sensor patch.

- class cTkSDHTactileSensorPatch

  A widget to display a single tactile sensor patch.

- class cTkSDHTactileSensorPatches

  Widget to display all tactile sensor patches of an SDH.

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "Simple tkInter elements to visualize tactile sensors of SDH"
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: tkdsa.py 4355 2009-05-04 17:17:39Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple dbg = sdh.dbg.tDBG(False, "cyan")
- def DisplayStyles

  Return a list of valid display styles.

9.29.1 Function Documentation

9.29.1.1 def sdh.tkdsa.DisplayStyles ( )

Return a list of valid display styles.

9.29.2 Variable Documentation

9.29.2.1 string sdh::tkdsa.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.29.2.2 string sdh::tkdsa.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.29.2.3 string sdh::tkdsa.__doc__ = "Simple tkInter elements to visualize tactile sensors of SDH"

9.29.2.4 string sdh::tkdsa.__url__ = "http://www.schunk.com"

9.29.2.5 string sdh::tkdsa.__version__ = "$Id: tkdsa.py 4355 2009-05-04 17:17:39Z Osswald2 $"

9.29.2.6 tuple sdh::tkdsa.dbg = sdh.dbg.tDBG(False, "cyan")

9.30 Package sdh.unit

Classes

- class cUnitConverter

  Unit conversion class.

Variables

Python specific variables

Some definitions that describe the module for python.

- string __doc__ = "Unit conversion class and objects."
- string __author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string __url__ = "http://www.schunk.com"
- string __version__ = "$Id: unit.py 4121 2009-02-11 19:30:06Z Osswald2 $"
- string __copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

Predefined unit conversion objecs

Some predefined cUnitConverter unit conversion objects to convert values between different unit systems. These can be used e.g. to convert angle values between degrees and radians, temperatures between degrees celsius and degrees fahrenheit or the like.

The cSDH class uses such 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.

- tuple uc_angle_degrees = cUnitConverter("angle", "degrees", u"\N{DEGREE SIGN}", 1.0, 0.0, 1)

  Default converter for angles (internal unit == external unit): degrees.

- tuple uc_angle_radians = cUnitConverter("angle", "radians", "rad", (2.0∗math.pi)/360.0, 0.0, 3)

  Converter for angles: external unit = radians.

- tuple uc_time_seconds = cUnitConverter("time", "seconds", "s", 1.0, 0.0, 3)

  Default converter for times (internal unit == external unit): seconds.

- tuple uc_time_milliseconds = cUnitConverter("time", "milliseconds", "ms", 1000.0, 0.0, 0)

  Converter for times: external unit = milliseconds.

- tuple uc_temperature_celsius = cUnitConverter("temparature", "degrees celsius", u"\N{DEGREE SIGN}C", 1.0, 0.0, 1)

  Default converter for temparatures (internal unit == external unit): degrees celsius.

- tuple uc_temperature_fahrenheit = cUnitConverter("temparature", "degrees fahrenheit", u"\N{DEGREE SIGN}F", 1.8, 32.0, 1)

  Converter for temperatures: external unit = degrees fahrenheit.

- tuple uc_angular_velocity_degrees_per_second = cUnitConverter("angular velocity", "degrees/second", u"\N{DEGREE SIGN}/s", 1.0, 0.0, 2)

Default converter for angular velocities (internal unit == external unit): degrees / second.

- tuple uc_angular_velocity_radians_per_second = cUnitConverter("angular velocity", "radians/second", "rad/s", (2.0∗math.pi)/360.0, 0.0, 4)

  Converter for angular velocieties: external unit = radians/second.

- tuple uc_angular_acceleration_degrees_per_second_squared = cUnitConverter("angular acceleration", "degrees/(second∗second)", u"\N{DEGREE SIGN}/s\N{SUPERSCRIPT TWO}", 1.0, 0.0, 1)

  Default converter for angular velocities (internal unit == external unit): degrees / (second ∗ second)

- tuple uc_angular_acceleration_radians_per_second_squared = cUnitConverter("angular acceleration", "radians/(second∗second)", u"rad/s\N{SUPERSCRIPT TWO}", (2.0∗math.pi)/360.0, 0.0, 3)

  Converter for angular velocieties: external unit = radians/(second∗second)

- tuple uc_motor_current_ampere = cUnitConverter("motor current", "Ampere", "A", 1.0, 0.0, 3)

  Default converter for motor current (internal unit == external unit): Ampere.

- tuple uc_motor_current_milliampere = cUnitConverter("motor current", "milli Ampere", "mA", 1000.0, 0.0, 0)

  Converter for motor current: external unit = milli Ampere.

- tuple uc_position_millimeter = cUnitConverter("position", "millimeter", "mm", 1.0, 0.0, 1)

  Default converter for position (internal unit == external unit): millimeter.

- tuple uc_position_meter = cUnitConverter("position", "meter", "m", 1/1000.0, 0.0, 4)

  Converter for position: external unit = meter.

9.30.1 Variable Documentation

9.30.1.1 string sdh::unit.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

9.30.1.2 string sdh::unit.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

9.30.1.3 string sdh::unit.__doc__ = "Unit conversion class and objects."

9.30.1.4 string sdh::unit.__url__ = "http://www.schunk.com"

9.30.1.5 string sdh::unit.__version__ = "$Id: unit.py 4121 2009-02-11 19:30:06Z Osswald2 $"

9.30.1.6 tuple sdh::unit.uc_angle_degrees = cUnitConverter("angle", "degrees", u"\N{DEGREE SIGN}", 1.0, 0.0, 1)

Default converter for angles (internal unit == external unit): degrees.

9.30.1.7 tuple sdh::unit.uc_angle_radians = cUnitConverter("angle", "radians", "rad", (2.0∗math.pi)/360.0, 0.0, 3)

Converter for angles: external unit = radians.

9.30.1.8 tuple sdh::unit.uc_angular_acceleration_degrees_per_second_squared = cUnitConverter("angular acceleration", "degrees/(second∗second)", u"\N{DEGREE SIGN}/s\N{SUPERSCRIPT TWO}", 1.0, 0.0, 1)

Default converter for angular velocities (internal unit == external unit): degrees / (second ∗ second)

9.30.1.9 tuple sdh::unit.uc_angular_acceleration_radians_per_second_squared = cUnitConverter("angular acceleration", "radians/(second∗second)", u"rad/s\N{SUPERSCRIPT TWO}", (2.0∗math.pi)/360.0, 0.0, 3)

Converter for angular velocieties: external unit = radians/(second∗second)

9.30.1.10 tuple sdh::unit.uc_angular_velocity_degrees_per_second = cUnitConverter("angular velocity", "degrees/second", u"\N{DEGREE SIGN}/s", 1.0, 0.0, 2)

Default converter for angular velocities (internal unit == external unit): degrees / second.

9.30.1.11 tuple sdh::unit.uc_angular_velocity_radians_per_second = cUnitConverter("angular velocity", "radians/second", "rad/s", (2.0*math.pi)/360.0, 0.0, 4)

Converter for angular velocieties: external unit = radians/second.

9.30.1.12 tuple sdh::unit.uc_motor_current_ampere = cUnitConverter("motor current", "Ampere", "A", 1.0, 0.0, 3)

Default converter for motor current (internal unit == external unit): Ampere.

9.30.1.13 tuple sdh::unit.uc_motor_current_milliampere = cUnitConverter("motor current", "milli Ampere", "mA", 1000.0, 0.0, 0)

Converter for motor current: external unit = milli Ampere.

9.30.1.14 tuple sdh::unit.uc_position_meter = cUnitConverter("position", "meter", "m", 1/1000.0, 0.0, 4)

Converter for position: external unit = meter.

9.30.1.15 tuple sdh::unit.uc_position_millimeter = cUnitConverter("position", "millimeter", "mm", 1.0, 0.0, 1)

Default converter for position (internal unit == external unit): millimeter.

9.30.1.16 tuple sdh::unit.uc_temperature_celsius = cUnitConverter("temparature", "degrees celsius", u"\N{DEGREE SIGN}C", 1.0, 0.0, 1)

Default converter for temparatures (internal unit == external unit): degrees celsius.

9.30.1.17 tuple sdh::unit.uc_temperature_fahrenheit = cUnitConverter("temparature", "degrees fahrenheit", u"\N{DEGREE SIGN}F", 1.8, 32.0, 1)

Converter for temperatures: external unit = degrees fahrenheit.

9.30.1.18 tuple sdh::unit.uc_time_milliseconds = cUnitConverter("time", "milliseconds", "ms", 1000.0, 0.0, 0)

Converter for times: external unit = milliseconds.

9.30.1.19 tuple sdh::unit.uc_time_seconds = cUnitConverter("time", "seconds", "s", 1.0, 0.0, 3)

Default converter for times (internal unit == external unit): seconds.

9.31 Package sdh.util

Classes

- class tMyOptionParser

  OptionParser with some default options already set: -d | --debug turn on debug (set options.debug flag) -v | --version print version and exit.

Functions

- def GetColor

  return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_-back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

- def Beep

  Do n console beeps with a delay of delay seconds.

- def GetClipboard

  Return content of clipboard (on cygwin/Windows)

- def SetClipboard

  Set content of clipboard to content (on cygwin/Windows)

- def WinpathToCygpath

  Return the cygwin path of the file with the windows path winpath.

- def error
- def Ziplen

  return a list containing tuples of elements and indexes of these elements Remark: this is like the std enumerate(l) with the elements in the tuples reversed

- def Call

  call function fun with arguments pars.

- def sgn

  return signum of v

- def GetDefineOrVariable

  Return value of C/C++ define "name" in header file "ifile" or of python variable "name" in python module "ifile".

- def GetProjectName

  Return name of project (extracted from header file release_file).

- def GetProjectRelease

  Return release of project (extracted from header file release_file).

- def RangeDefToList

  return a list of indexes according to a range definition string e.g.

- def GetPersistantDict

  Dictionary that stores objects persistently using shelve.

Variables

- string __doc__

  The docstring describing the purpose of the script:

9.31.1 Function Documentation

9.31.1.1 def sdh.util.Beep (*n* = `1`, *delay* = `0.2`)

Do n console beeps with a delay of delay seconds.

9.31.1.2 def sdh.util.Call (*fun*, *pars*)

call function fun with arguments pars.

- pars = None : call fun()
- pars = SomeType : call fun(pars)
- pars = tuple : call fun(*pars)

9.31.1.3 def sdh.util.error (*args*)

9.31.1.4 def sdh.util.GetClipboard ( )

Return content of clipboard (on cygwin/Windows)

9.31.1.5 def sdh.util.GetColor (*c*)

return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_-back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

If the environment variable "SDH_NO_COLOR" is set then "" is returned. If the environment variable "OS" is WIN∗ or Win∗ and "OSTYPE" is not "cygwin" then "" is returned. (to prevent color output on windows consoles which cannot handle it). If the color is not found in the list of known colors then the string "" is returned.

9.31.1.6 def sdh.util.GetDefineOrVariable (*ifile, name*)

Return value of C/C++ define "name" in header file "ifile" or of python variable "name" in python module "ifile".

9.31.1.7 def sdh.util.GetPersistantDict (*name* = `None`, *path* = `None`, *cdbg* = `None`)

Dictionary that stores objects persistently using shelve.

If you want to be able to generate standalone exes with py2exe you must add the following line to your script according to `http://myblog.vindaloo.com/?p=129` import anydbm, dbhash

9.31.1.8 def sdh.util.GetProjectName (*release_file*)

Return name of project (extracted from header file release_file).

The code below uses a regular expression to extracts the value of the C preprocessor macro define or the definition of a variable named PROJECT_NAME. The extracted value can be:

- Used by doxygen as project name.
- Used as base name of the generated pdf documentation files.
- Used as name of project directory when installing.

9.31.1.9 def sdh.util.GetProjectRelease (*release_file*)

Return release of project (extracted from header file release_file).

The code below uses a regular expression to extracts the value of the C preprocessor macro define or the definition of a variable named PROJECT_RELEASE. The extracted value can be:

- Used by doxygen as project release.

- Used as name of release directory when installing.

9.31.1.10 def sdh.util.RangeDefToList (*range_definition*, *max* = `1000`)

return a list of indexes according to a range definition string e.g.

"1" => [1], "1,2,4" => [1,2,4], "3-6" => [3,4,5,6])

9.31.1.11 def sdh.util.SetClipboard (*content*)

Set content of clipboard to content (on cygwin/Windows)

9.31.1.12 def sdh.util.sgn (*v*)

return signum of v

9.31.1.13 def sdh.util.WinpathToCygpath (*winpath*)

Return the cygwin path of the file with the windows path winpath.

Will convert "c:\\bla\\blu.bli" to "/cygdrive/c/bla/blu.bli"

9.31.1.14 def sdh.util.Ziplen (*l*)

return a list containing tuples of elements and indexes of these elements Remark: this is like the std enumerate(l) with the elements in the tuples reversed

9.31.2 Variable Documentation

9.31.2.1 string sdh::util.__doc__

Initial value:

```
'''
util.py:       This is a python module. It is meant to be imported by other modules and scripts.
Brief:         A collection of generally usefull python functions and classes:
               - GetColor    : return console color code
               - Beep        : beep on console
               - GetClipboard: Return content of clipboard (on cygwin/Windows)
               - SetClipboard: Set content of clipboard to content (on cygwin/Windows)
               - WinpathToCygpath: return cygwin path from windows path
               - tMyOptionParser: OptionParser with some defaults set (like -d -v)

                - error          : print on stderr
                - Ziplen         : return a list containing tuples of elements and indexes of these elements
                - call           : call function with 0,1,n arguments
```

```
               - sgn             : return signum of numeric value
               - GetDefineOrVariable : extract value of define from header file or variable from python file
               - GetProjectName      : extract value of PROJECT_NAME from header file or variable from python file
               - GetProjectRelease   : extract value of PROJECT_RELEASE from header file or variable from python file
               - RangeDefToList      : convert a range definition description to a list of indices, like "1-3,5" => [1,2,3,5]

Author:       Dirk Osswald <dirk_osswald@web.de>
Date:         2006-04-07
CVS-revision: $Id: util.py 12281 2014-09-30 07:44:33Z Osswald2 $
'''
```

The docstring describing the purpose of the script:

9.32 Package sdh.utils

Classes

- class bool

  Introduced in 2.3.

- class BaseSet

  sets module introduced in 2.3

- class frozenset
- class set
- class DefaultDict

  Dictionary with a default value for unknown keys.

- class Struct

  Create an instance with argument=value slots.

- class Queue

  Queue is an abstract class/interface.

- class FIFOQueue

  A First-In-First-Out Queue.

- class PriorityQueue

  A queue in which the minimum (or maximum) element (as determined by f and order) is returned first.

Functions

- def sum

  Introduced in 2.3.
- def enumerate

  Return an iterator that enumerates pairs of (i, c[i]).
- def reversed

  Iterate over x in reverse order.
- def sorted

  Copy seq and sort and return it.
- def Dict

  Create a dict out of the argument=value arguments.
- def update

  Update a dict; or an object with slots; according to entries.
- def removeall

  Return a copy of seq (or string) with all occurences of item removed.
- def unique

  Remove duplicate elements from seq.
- def product

  Return the product of the numbers.
- def count_if

  Count the number of elements of seq for which the predicate is true.
- def find_if

  If there is an element of seq that satisfies predicate; return it.
- def every

  True if every element of seq satisfies predicate.
- def some

  If some element x of seq satisfies predicate(x), return predicate(x).
- def isin

  Like (elt in seq), but compares with is, not ==.
- def argmin

  Return an element with lowest fn(seq[i]) score; tie goes to first one.

- def argmin_list

  Return a list of elements of seq[i] with the lowest fn(seq[i]) scores.

- def argmin_random_tie

  Return an element with lowest fn(seq[i]) score; break ties at random.

- def argmax

  Return an element with highest fn(seq[i]) score; tie goes to first one.

- def argmax_list

  Return a list of elements of seq[i] with the highest fn(seq[i]) scores.

- def argmax_random_tie
- def histogram

  Return a list of (value, count) pairs, summarizing the input values.

- def log2

  Base 2 logarithm.

- def mode

  Return the most common value in the list of values.

- def median

  Return the middle value, when the values are sorted.

- def mean

  Return the arithmetic average of the values.

- def stddev

  The standard deviation of a set of values.

- def dotproduct

  Return the sum of the element-wise product of vectors x and y.

- def vector_add

  Component-wise addition of two vectors.

- def probability
- def num_or_str

  The argument is a string; convert to a number if possible, or strip it.

- def normalize

  Multiply each number by a constant such that the sum is 1.0 (or total).

- def turn_right

- def turn_left
- def distance
- def distance2
- def clip

  Return vector, except if any element is less than the corresponding value of lowest or more than the corresponding value of highest, clip to those values.

- def printf

  Format args with the first argument as format string, and write.

- def caller

  Return the name of the calling function n levels up in the frame stack.

- def memoize

  Memoize fn: make it remember the computed value for any argument list.

- def if_

  Like C++ and Java's (test ? result : alternative), except both result and alternative are always evaluated.

- def name
- def isnumber
- def issequence
- def print_table

  Print a list of lists as a table, so that columns line up nicely.

- def Stack

  Return an empty list, suitable as a Last-In-First-Out Queue.

Variables

- float infinity = 1.0e400
- list orientations = [(1,0), (0, 1), (-1, 0), (0, -1)]

  OK, the following are not as widely useful utilities as some of the other functions here, but they do show up wherever we have 2D grids: Wumpus and Vacuum worlds, TicTacToe and Checkers, and markov decision Processes.

- dictionary Fig = { }

  Fig: The idea is we can define things like Fig[3,10] later.

9.32.1 Function Documentation

9.32.1.1 def sdh.utils.argmax (*seq*, *fn*)

Return an element with highest fn(seq[i]) score; tie goes to first one.

>>> argmax(['one', 'to', 'three'], len) 'three'

9.32.1.2 def sdh.utils.argmax_list (*seq*, *fn*)

Return a list of elements of seq[i] with the highest fn(seq[i]) scores.

>>> argmax_list(['one', 'three', 'seven'], len) ['three', 'seven']

9.32.1.3 def sdh.utils.argmax_random_tie (*seq*, *fn*)

9.32.1.4 def sdh.utils.argmin (*seq*, *fn*)

Return an element with lowest fn(seq[i]) score; tie goes to first one.

>>> argmin(['one', 'to', 'three'], len) 'to'

9.32.1.5 def sdh.utils.argmin_list (*seq*, *fn*)

Return a list of elements of seq[i] with the lowest fn(seq[i]) scores.

>>> argmin_list(['one', 'to', 'three', 'or'], len) ['to', 'or']

9.32.1.6 def sdh.utils.argmin_random_tie (*seq*, *fn*)

Return an element with lowest fn(seq[i]) score; break ties at random.

Thus, for all s,f: argmin_random_tie(s, f) in argmin_list(s, f)

9.32.1.7 def sdh.utils.caller (*n* = `1`)

Return the name of the calling function n levels up in the frame stack.

>>> caller(0) 'caller' >>> def f(): ... return caller() >>> f() 'f'

9.32.1.8 def sdh.utils.clip (*vector*, *lowest*, *highest*)

Return vector, except if any element is less than the corresponding value of lowest or more than the corresponding value of highest, clip to those values.

>>> clip((-1, 10), (0, 0), (9, 9)) (0, 9)

9.32.1.9 def sdh.utils.count_if (*predicate, seq*)

Count the number of elements of seq for which the predicate is true.

>>> count_if(callable, [42, None, max, min]) 2

9.32.1.10 def sdh.utils.Dict (*entries*)

Create a dict out of the argument=value arguments.

>>> Dict(a=1, b=2, c=3) {'a': 1, 'c': 3, 'b': 2}

9.32.1.11 def sdh.utils.distance (*ax, ay, bx, by*)

9.32.1.12 def sdh.utils.distance2 (*ax, ay, bx, by*)

9.32.1.13 def sdh.utils.dotproduct (*X, Y*)

Return the sum of the element-wise product of vectors x and y.

>>> dotproduct([1, 2, 3], [1000, 100, 10]) 1230

9.32.1.14 def sdh.utils.enumerate (*collection*)

Return an iterator that enumerates pairs of (i, c[i]).

PEP 279. >>> list(enumerate('abc')) [(0, 'a'), (1, 'b'), (2, 'c')]

9.32.1.15 def sdh.utils.every (*predicate, seq*)

True if every element of seq satisfies predicate.

>>> every(callable, [min, max]) 1 >>> every(callable, [min, 3]) 0

9.32.1.16 def sdh.utils.find_if (*predicate, seq*)

If there is an element of seq that satisfies predicate; return it.

>>> find_if(callable, [3, min, max]) <built-in function min> >>> find_if(callable, [1, 2, 3])

9.32.1.17 def sdh.utils.histogram (*values, mode =* `0`*, bin_function =* `None`)

Return a list of (value, count) pairs, summarizing the input values.

Sorted by increasing value, or if mode=1, by decreasing count. If bin_function is given, map it over values first.

9.32.1.18 def sdh.utils.if_ (*test, result, alternative*)

Like C++ and Java's (test ? result : alternative), except both result and alternative are always evaluated.

However, if either evaluates to a function, it is applied to the empty arglist, so you can delay execution by putting it in a lambda. >>> if_(2 + 2 == 4, 'ok', lambda: expensive_computation()) 'ok'

9.32.1.19 def sdh.utils.isin (*elt, seq*)

Like (elt in seq), but compares with is, not ==.

>>> e = []; isin(e, [1, e, 3]) True >>> isin(e, [1, [ ], 3]) False

9.32.1.20 def sdh.utils.isnumber (*x*)

9.32.1.21 def sdh.utils.issequence (*x*)

9.32.1.22 def sdh.utils.log2 (*x*)

Base 2 logarithm.

>>> log2(1024) 10.0

9.32.1.23 def sdh.utils.mean (*values*)

Return the arithmetic average of the values.

9.32.1.24 def sdh.utils.median (*values*)

Return the middle value, when the values are sorted.

If there are an odd number of elements, try to average the middle two. If they can't be averaged (e.g. they are strings), choose one at random. >>> median([10, 100, 11]) 11 >>> median([1, 2, 3, 4]) 2.5

9.32.1.25 def sdh.utils.memoize (*fn, slot =* `None`)

Memoize fn: make it remember the computed value for any argument list.

If slot is specified, store result in that slot of first argument. If slot is false, store results in a dictionary.

9.32.1.26 def sdh.utils.mode (*values*)

Return the most common value in the list of values.

>>> mode([1, 2, 3, 2]) 2

9.32.1.27 def sdh.utils.name (*object*)

9.32.1.28 def sdh.utils.normalize (*numbers, total =* `1.0`)

Multiply each number by a constant such that the sum is 1.0 (or total).

>>> normalize([1,2,1]) [0.25, 0.5, 0.25]

9.32.1.29 def sdh.utils.num_or_str (*x*)

The argument is a string; convert to a number if possible, or strip it.

>>> num_or_str('42') 42 >>> num_or_str(' 42x ') '42x'

9.32.1.30 def sdh.utils.print_table (*table, header =* `None`*, sep =* `' '`*, numfmt =* `'%g'`)

Print a list of lists as a table, so that columns line up nicely.

header, if specified, will be printed as the first row. numfmt is the format for all numbers; you might want e.g. '6.2f'. (If you want different formats in differnt columns, don't use print_table.) sep is the separator between columns.

9.32.1.31 def sdh.utils.printf (*format, args*)

Format args with the first argument as format string, and write.

Return the last arg, or format itself if there are no args.

9.32.1.32 def sdh.utils.probability (*p*)

9.32.1.33 def sdh.utils.product (*numbers*)

Return the product of the numbers.

>>> product([1,2,3,4]) 24

9.32.1.34 def sdh.utils.removeall (*item, seq*)

Return a copy of seq (or string) with all occurences of item removed.

>>> removeall(3, [1, 2, 3, 3, 2, 1, 3]) [1, 2, 2, 1] >>> removeall(4, [1, 2, 3]) [1, 2, 3]

9.32.1.35 def sdh.utils.reversed (*seq*)

Iterate over x in reverse order.

>>> list(reversed([1,2,3])) [3, 2, 1]

9.32.1.36 def sdh.utils.some (*predicate*, *seq*)

If some element x of seq satisfies predicate(x), return predicate(x).

>>> some(callable, [min, 3]) 1 >>> some(callable, [2, 3]) 0

9.32.1.37 def sdh.utils.sorted (*seq*, *cmp* = `None`, *key* = `None`, *reverse* = `False`)

Copy seq and sort and return it.

>>> sorted([3, 1, 2]) [1, 2, 3]

9.32.1.38 def sdh.utils.Stack ( )

Return an empty list, suitable as a Last-In-First-Out Queue.

9.32.1.39 def sdh.utils.stddev (*values*, *meanval* = `None`)

The standard deviation of a set of values.

Pass in the mean if you already know it.

9.32.1.40 def sdh.utils.sum (*seq*, *start* = `0`)

Introduced in 2.3.

Sum the elements of seq. >>> sum([1, 2, 3]) 6

9.32.1.41 def sdh.utils.turn_left (*orientation*)

9.32.1.42 def sdh.utils.turn_right (*orientation*)

9.32.1.43 def sdh.utils.unique (*seq*)

Remove duplicate elements from seq.

Assumes hashable elements. >>> unique([1, 2, 3, 2, 1]) [1, 2, 3]

9.32.1.44 def sdh.utils.update (*x*, *entries*)

Update a dict; or an object with slots; according to entries.

>>> update({'a': 1}, a=10, b=20) {'a': 10, 'b': 20} >>> update(Struct(a=1), a=10, b=20) Struct(a=10, b=20)

9.32.1.45 def sdh.utils.vector_add (*a*, *b*)

Component-wise addition of two vectors.

>>> vector_add((0, 1), (8, 9)) (8, 10)

9.32.2 Variable Documentation

9.32.2.1 dictionary sdh::utils.Fig = {}

Fig: The idea is we can define things like Fig[3,10] later.

Alas, it is Fig[3,10] not Fig[3.10], because that would be the same as Fig[3.1]

9.32.2.2 float sdh::utils.infinity = 1.0e400

9.32.2.3 list sdh::utils.orientations = [(1,0), (0, 1), (-1, 0), (0, -1)]

OK, the following are not as widely useful utilities as some of the other functions here, but they do show up wherever we have 2D grids: Wumpus and Vacuum worlds, TicTacToe and Checkers, and markov decision Processes.

9.33 Package setup

Functions

- def Pathify

  join dirs (list of directory names/files) to a list of paths using the right path separator

Variables

- tuple sdh_locals = dict()
- tuple sdh_globals = dict()
- list doc_files
- list guidat_files
- tuple src_rel_paths = map(lambda n: Pathify(r, n)[0], f)
- tuple target_path
- list version = sdh_locals["PROJECT_RELEASE"]
- string description = 'sdh: the python package to access an SDH (SCHUNK Dexterous Hand)'
- string author = 'Dirk Osswald'
- string author_email = 'dirk.osswald@de.schunk.com'
- string url = 'http://www.schunk.com/'
- string long_description
- list packages = ['sdh']
- tuple scripts = Pathify('demo', 'demo-simple.py')
- data_files = doc_files+guidat_files

9.33.1 Function Documentation

9.33.1.1 def setup.Pathify (*dirs*)

join dirs (list of directory names/files) to a list of paths using the right path separator

9.33.2 Variable Documentation

9.33.2.1 string setup.author = 'Dirk Osswald'

9.33.2.2 string setup.author_email = 'dirk.osswald@de.schunk.com'

9.33.2.3 setup.data_files = doc_files+guidat_files

9.33.2.4 string setup.description = 'sdh: the python package to access an SDH (SCHUNK Dexterous Hand)'

9.33.2.5 list setup.doc_files

Initial value:

```
[ ( 'share/doc/%s' %  sdh_locals[ "PROJECT_NAME" ],         # target dir
                Pathify('doc', 'index-sdhlibrary-python.html')          # list of files to install in target dir
                + Pathify('doc', 'SDHLibrary-python-external.pdf')
                + Pathify('..', '..', 'software', 'doc', 'SDH2_configuration-and-update.pdf')
#                + Pathify('doc', 'Inbetriebnahme.doc')
                )
              ]
```

9.33.2.6 list setup.guidat_files

Initial value:

```
[ ('scripts',  # target dir
                  Pathify( 'demo', 'positions.guidat' )
                  + Pathify( 'demo', 'warmup.guidat' )
                  )
              ]
```

9.33.2.7 string setup.long_description

Initial value:

```
'''
This is a python package to access an SDH (SCHUNK Dexterous Hand).
It provides a class interface to access the functionality of the SDH
and some example scripts that demonstrate its use.
For details see the included documentation in html- or pdf-format.
'''
```

9.33.2.8 list setup.packages = ['sdh']

9.33.2.9 tuple setup.scripts = Pathify('demo', 'demo-simple.py')

9.33.2.10 tuple setup.sdh_globals = dict()

9.33.2.11 tuple setup.sdh_locals = dict()

9.33.2.12 tuple setup.src_rel_paths = map(lambda n: Pathify(r, n)[0], f)

9.33.2.13 tuple setup.target_path

Initial value:

```
r.replace( os.path.join( 'doc', ''),
                              os.path.join( 'share', 'doc', sdh_locals[ "PROJECT_N
      AME" ], '' ) )
```

9.33.2.14 string setup.url = 'http://www.schunk.com/'

9.33.2.15 list setup.version = sdh_locals["PROJECT_RELEASE"]

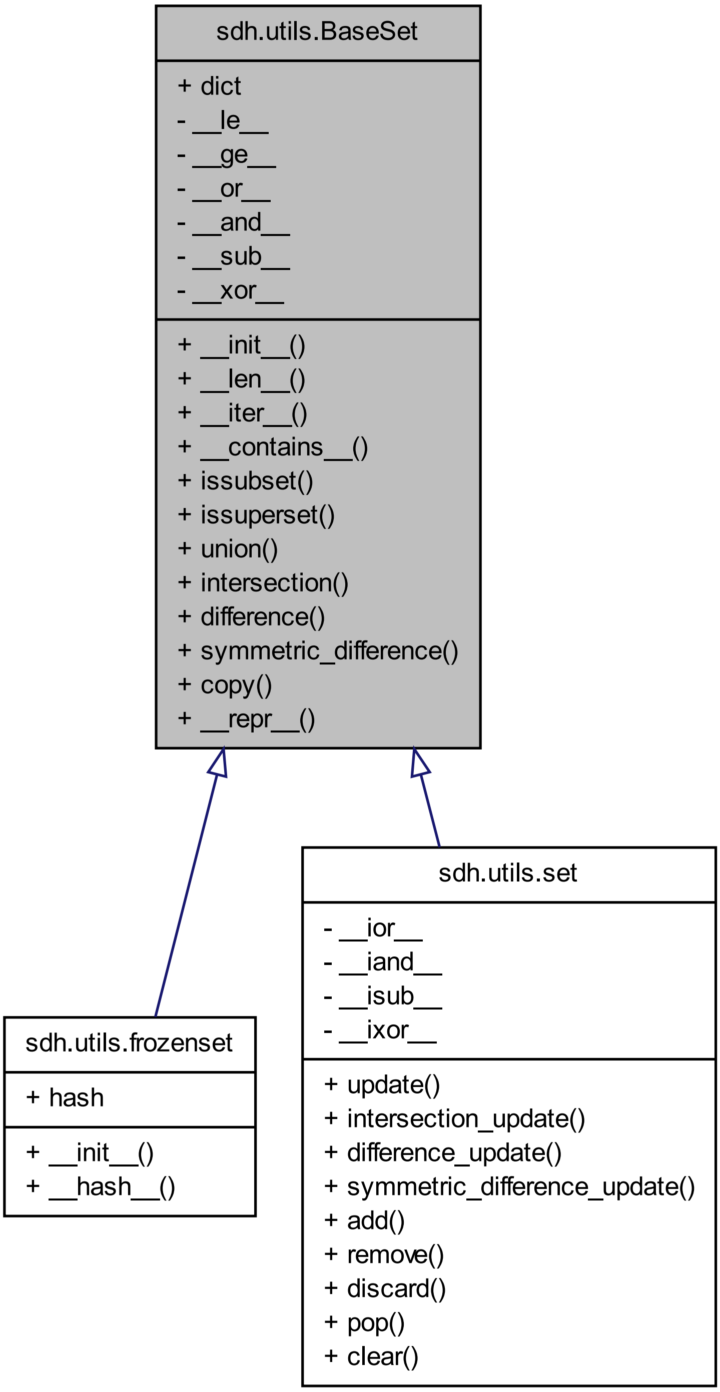
Chapter 10

Class Documentation

10.1 sdh.utils.BaseSet Class Reference

sets module introduced in 2.3

Inheritance diagram for sdh.utils.BaseSet:



Public Member Functions

- def __init__
- def __len__
- def __iter__
- def __contains__
- def issubset
- def issuperset
- def union
- def intersection
- def difference
- def symmetric_difference
- def copy
- def __repr__

Public Attributes

- dict

10.1.1 Detailed Description

sets module introduced in 2.3

10.1.2 Constructor & Destructor Documentation

10.1.2.1 def sdh.utils.BaseSet.__init__ (*self*, *elements* = [])

Reimplemented in sdh.utils.frozenset.

10.1.3 Member Function Documentation

10.1.3.1 def sdh.utils.BaseSet.__contains__ (*self*, *element*)

10.1.3.2 def sdh.utils.BaseSet.__iter__ (*self*)

10.1.3.3 def sdh.utils.BaseSet.__len__ (*self*)

10.1.3.4 def sdh.utils.BaseSet.__repr__ (*self*)

10.1.3.5 def sdh.utils.BaseSet.copy (*self*)

10.1.3.6 def sdh.utils.BaseSet.difference (*self*, *other*)

10.1.3.7 def sdh.utils.BaseSet.intersection (*self*, *other*)

10.1.3.8 def sdh.utils.BaseSet.issubset (*self*, *other*)

10.1.3.9 def sdh.utils.BaseSet.issuperset (*self*, *other*)

10.1.3.10 def sdh.utils.BaseSet.symmetric_difference (*self*, *other*)

10.1.3.11 def sdh.utils.BaseSet.union (*self*, *other*)

10.1.4 Member Data Documentation

10.1.4.1 sdh.utils.BaseSet.dict

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

- sdh/utils.py

10.2 sdh.utils.bool Class Reference

Introduced in 2.3.

Public Member Functions

- def __init__
- def __int__
- def __repr__

Public Attributes

- val

10.2.1 Detailed Description

Introduced in 2.3.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 def sdh.utils.bool.__init__ (*self*, *val*)

10.2.3 Member Function Documentation

10.2.3.1 def sdh.utils.bool.__int__ (*self*)

10.2.3.2 def sdh.utils.bool.__repr__ (*self*)

10.2.4 Member Data Documentation

10.2.4.1 sdh.utils.bool.val

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

- sdh/utils.py

10.3 sdh.dsa.cDSA Class Reference

Interface class to access the DSACON32m, the tactile sensor controller of the SDH.

Public Member Functions

- def __init__

  Constructor of cDSA class.

- def read
- def write
- def timeout
- def timeout
- def GetTimeoutRS232
- def SetTimeoutRS232
- def GetTimeoutTCP
- def SetTimeoutTCP
- def FlushInput

  Cleanup communication line: read all available bytes with timeout_s_first timeout in s for first byte and timeout_s_subsequent timeoutin s for subsequent bytes.

- def CheckErrorCode

  Check error_code error_code.

- def Close

  Close connection to remote DSACON32m controller in the SDH.

- def CleanCommunicationLine

  Clean up the communication line by reading up to 1000 bytes with timeout 0, i.e.

- def SetFramerate

  Set the framerate for querying full frames.

- def SetFramerateRetries

  Set the framerate for querying full frames.

- def ReadFrame

  read and parse a full frame response from remote DSA

- def QueryControllerInfo

  Send command to respond controller info to remote DSA.

- def QuerySensorInfo

  Send command to respond sensor info to remote DSA.

- def QueryMatrixInfo

  Send command to respond matrix info to remote DSA.

- def SetMatrixSensitivity

  Send command to set matrix sensitivity to sensitivity as float [0.0 .

- def GetMatrixSensitivity

  Send command to get matrix sensitivity.

- def SetMatrixThreshold

  Send command to set matrix threshold to threshold as integer [0 .

- def GetMatrixThreshold

  Send command to get matrix threshold.

- def GetTexel

  Return texel value at column x row y of frame.

- def PrintMessage

  Debug function: print bytes of message.

- def PrintFrame

  Debug function: print frame.

- def StartUpdater

  Make remote DSA send frames with framerate.

- def GetMatrixIndex

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

- def GetAgeOfFrame

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

- def GetContactArea

  *Return contact area in mm*mm for finger(s) fi and sensor part(s) part.*

- def GetContactForce

  Return a tuple (force,cog_x,cog_y,area) of contact force and center of gravity and contact area of that force for finger fi and sensor part.

Public Attributes

- com
- port
- GetTimeout
- SetTimeout
- acquiring_single_frame

  flag, true if user requested acquiring of a single frame.

- timeout

  The read timeout for accessing the interface in seconds.

- controller_info

  A structure holding info about the remote DSACON32m controller.

- sensor_info

  A structure holding info about the remote DSACON32m sensor.

- matrix_info

  A list of structures holding info about the remote tactile sensors connected to the DSACON32m.

- texel_offset

  A list of texel offsets.

- frame

  A structure containing the last tactile sensor frame read from the SDH.

- all_fingers

  A list of all the finger indices of the SDH.

- all_parts

  A list of all the tactile sensor part indices of a finger of the SDH.

- contact_area_cell_threshold

  threshold of texel cell value for detecting contacts with GetContactArea

- contact_force_cell_threshold

  threshold of texel cell value for detecting forces with GetContactForce

- force_factor

  additional calibration factor for forces in GetContactForce

- calib_pressure

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

- calib_voltage

  see calib_pressure

- read_another

  flag, if True then the ReadFrame() function will read tactile sensor frames until a timeout occurs.

Static Public Attributes

- list error_codes

  A list of triples (error code (int), error code name (string), error code description (string)) These are the error codes reported by the remote DSACON32m controller.

- tuple eDSAPacketID

  A dictionary of packet ID to number mappings.

10.3.1 Detailed Description

Interface class to access the DSACON32m, the tactile sensor controller of the SDH.

Bug

SCHUNK-internal bugzilla ID: Bug 983
With SDHLibrary-Python < 0.0.2.1 communication to the DSACON32m tactile sensor controller within the SDH was not established correctly in some cases. The default baudrate of the RS232 port was not set correctly unless specified explicitly.
=> Resolved in SDHLibrary-Python 0.0.2.1

10.3.2 Constructor & Destructor Documentation

10.3.2.1 def sdh.dsa.cDSA.__init__ (*self*, *debug_level* = `0`, *port* = `None`, *baudrate* = `115200`, *bytesize* = `8`, *parity* = `'N'`, *stopbits* = `1`, *timeout* = `1`, *xonxoff* = `0`, *rtscts* = `0`, *writeTimeout* = `None`, *dsrdtr* = `None`, *debug_output* = `sys.stderr`)

Constructor of cDSA class.

This constructs a cDSA object to communicate with the remote DSACON32m controller within the SDH.

The connection is opened and established, and the sensor_info, controller_info 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

self	- the instance of the class that this function operates on (the "object")
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 communication to use • a single number like 0 for an RS232 port (port 0 = ttyS0 = COM1, port 1 = ttyS1 = COM2, • or a device name like "/dev/ttyUSB0" for the corresponding RS232 port, • or a IP_OR_HOSTNAME:PORT for a TCP connection to that numeric IPv4 address or hostname
baudrate	- the baudrate to use. Leave this at the default 115200 bit/s. A value of 0 will use the default.
bytesize	- the size in bits of one byte to transfer. Leave this at the default 8 bit / byte.
parity	- the parity to use for transfer. Leave this at the default 'N' for no parity.
stopbits	- the number of stop bits to use for transfer. Leave this at the default 1.
timeout	- the timeout in seconds to use for transfer.
xonxoff	- the Xon/Xoff settung to use for transfer. Leave this at the default 0.
rtscts	- the RTS/CTS setting to use for transfer. Leave this at the default 0.
writeTimeout	- the write timeout to use for transfer. Leave this at the default None.
dsrdtr	- the DSR/DTR setting to use for transfer. Leave this at the default None.
debug_output	- a file like object where debug output is sent to, if enabled. Default is stderr.

10.3.3 Member Function Documentation

10.3.3.1 def sdh.dsa.cDSA.CheckErrorCode (*self*, *error_code*, *msg* = " ")

Check error_code *error_code*.

Raise a cDSAError including *msg* in case of error

10.3.3.2 def sdh.dsa.cDSA.CleanCommunicationLine (*self*)

Clean up the communication line by reading up to 1000 bytes with timeout 0, i.e.

return at once ignoring anything that is available now.

10.3.3.3 def sdh.dsa.cDSA.Close (*self*)

Close connection to remote DSACON32m controller in the SDH.

Tries to reset the framerate to 0 to stop the DSACON32m from sending before closing

10.3.3.4 def sdh.dsa.cDSA.FlushInput (*self*, *timeout_s_first*, *timeout_s_subsequent*)

Cleanup communication line: read all available bytes with *timeout_s_first* timeout in s for first byte and *timeout_s_subsequent* timeoutin s for subsequent bytes.

Parameters

timeout_s_first	- timeout in s for first byte
timeout_s_subsequent	- timeout in s for subsequent bytes

The push mode of the DSACON32m must be switched off on call since else the method will not return.

10.3.3.5 def sdh.dsa.cDSA.GetAgeOfFrame (*self*, *frame* = `None`)

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

10.3.3.6 def sdh.dsa.cDSA.GetContactArea (*self*, *fi* = All, *part* = All, *frame* = `None`)

Return contact area in mm∗mm for finger(s) fi and sensor part(s) part.

10.3.3.7 def sdh.dsa.cDSA.GetContactForce (*self*, *fi*, *part*, *frame* = None)

Return a tuple (force,cog_x,cog_y,area) of contact force and center of gravity and contact area of that force for finger fi and sensor part.

force is in N, cog_x,cog_ in mm, area in mm∗mm.

10.3.3.8 def sdh.dsa.cDSA.GetMatrixIndex (*self*, *fi*, *part*)

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

10.3.3.9 def sdh.dsa.cDSA.GetMatrixSensitivity (*self*, *matrix_no*)

Send command to get matrix sensitivity.

Returns sensitivities of matrix no *matrix_no*.

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

Raises a cDSAError in case of invalid responses from the remote DSACON32m controller.

Bug

With DSACON32m firmware R218 and before this did not work, instead the factory default (0.5) was always reported
=> Resolved in DSACON32m firmware R268

10.3.3.10 def sdh.dsa.cDSA.GetMatrixThreshold (*self*, *matrix_no*)

Send command to get matrix threshold.

Returns threshold of matrix no *matrix_no*.

A struct is returned containing the members *error_code* - see DSACON32 Command Set Reference Manual *threshold* - the currently set threshold as integer [0 .. 4095] (0 is minimum, 4095 is maximum threshold)

Raises a cDSAError in case of invalid responses from the remote DSACON32m controller.

Remarks

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

10.3.3.11 def sdh.dsa.cDSA.GetTexel (*self*, *m*, *x*, *y*)

Return texel value at column x row y of frame.

10.3.3.12 def sdh.dsa.cDSA.GetTimeoutRS232 (*self*)

10.3.3.13 def sdh.dsa.cDSA.GetTimeoutTCP (*self*)

10.3.3.14 def sdh.dsa.cDSA.PrintFrame (*self*)

Debug function: print frame.

10.3.3.15 def sdh.dsa.cDSA.PrintMessage (*self*, *message*)

Debug function: print bytes of message.

10.3.3.16 def sdh.dsa.cDSA.QueryControllerInfo (*self*)

Send command to respond controller info to remote DSA.

Read and parse the response from the remote DSA.

This is already done by the constructor and cached in self.controller_info

10.3.3.17 def sdh.dsa.cDSA.QueryMatrixInfo (*self*, *matrix_no*)

Send command to respond matrix info to remote DSA.

Read and parse the response from the remote DSA.

This is already done by the constructor and cached in the self.matrix_info list

10.3.3.18 def sdh.dsa.cDSA.QuerySensorInfo (*self*)

Send command to respond sensor info to remote DSA.

Read and parse the response from the remote DSA.

This is already done by the constructor and cached in self.sensor_info

10.3.3.19 def sdh.dsa.cDSA.read (*self*, *n*)

10.3.3.20 def sdh.dsa.cDSA.ReadFrame (*self*)

read and parse a full frame response from remote DSA

10.3.3.21 def sdh.dsa.cDSA.SetFramerate (*self*, *framerate*, *do_RLE* = `True`, *do_data_acquisition* = `True`)

Set the *framerate* for querying full frames.

Parameters

self	- the instance of the class that this function operates on (the "object")
framerate	- rate of frames.
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

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

Bug

SCHUNK-internal bugzilla ID: Bug 680
With DSACON32m firmware R276 and after and SDHLibrary-Python v0.0.1.19 and before stopping of the sending did not work.
=> Resolved in SDHLibrary-Python v0.0.1.20

Bug

SCHUNK-internal bugzilla ID: Bug 680
With DSACON32m firmware before R276 and SDHLibrary-Python v0.0.1.20 and before acquiring of single frames did not work
=> Resolved in SDHLibrary-Python v0.0.1.21

Bug

SCHUNK-internal bugzilla ID: Bug 703
With DSACON32m firmware R288 and before and SDHLibrary-Python v0.0.2.1 and before tactile sensor frames could not be read reliably in single frame mode.
=> Resolved in DSACON32m firmware 2.9.0.0
=> Resolved in SDHLibrary-Python v0.0.2.1 with workaround for older DSACON32m firmwares

10.3.3.22 def sdh.dsa.cDSA.SetFramerateRetries (*self*, *framerate*, *do_RLE* = `True`, *do_data_acquisition* = `True`, *retries* = `0`, *ignore_exceptions* = `False`)

Set the *framerate* for querying full frames.

Parameters

self	- the instance of the class that this function operates on (the "object")
framerate	- rate of frames
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

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

Bug

With DSACON32m firmware R276 and after and SDHLibrary-Python v0.0.1.19 and before stopping of the sending did not work.
=> Resolved in SDHLibrary-Python v0.0.1.20

Bug

With DSACON32m firmware before R276 and SDHLibrary-Python v0.0.1.20 and before acquiring of single frames did not work
=> Resolved in SDHLibrary-C++ v0.0.1.21

10.3.3.23 def sdh.dsa.cDSA.SetMatrixSensitivity (*self*, *matrix_no*, *sensitivity*, *do_all_matrices* = `False`, *do_reset* = `False`, *do_persistent* = `False`)

Send command to set matrix sensitivity to *sensitivity* as float [0.0 .

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

Warning

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

Raises a cDSAError in case of invalid responses from the remote DSACON32m controller.

Remarks

Setting the matrix sensitivity persistently is only possible if the DSACON32m firmware is R268 or above.

10.3.3.24 def sdh.dsa.cDSA.SetMatrixThreshold (*self*, *matrix_no*, *threshold*, *do_all_matrices* = `False`, *do_reset* = `False`, *do_persistent* = `False`)

Send command to set matrix threshold to *threshold* as integer [0 .

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

Warning

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

Raises a cDSAError in case of invalid responses from the remote DSACON32m controller.

Remarks

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

10.3.3.25 def sdh.dsa.cDSA.SetTimeoutRS232 (*self*, *v*)

10.3.3.26 def sdh.dsa.cDSA.SetTimeoutTCP (*self*, *v*)

10.3.3.27 def sdh.dsa.cDSA.StartUpdater (*self*, *framerate*, *do_RLE* = `True`)

Make remote DSA send frames with framerate.

Create a thread that updates self.frame continuously (guarded by self.semaphore).

10.3.3.28 def sdh.dsa.cDSA.timeout (*self*, *value*)

10.3.3.29 def sdh.dsa.cDSA.timeout (*self*)

10.3.3.30 def sdh.dsa.cDSA.write (*self*, *s*)

10.3.4 Member Data Documentation

10.3.4.1 sdh.dsa.cDSA.acquiring_single_frame

flag, true if user requested acquiring of a single frame.

Needed for DSACON32m firmware-bug workaround.

10.3.4.2 sdh.dsa.cDSA.all_fingers

A list of all the finger indices of the SDH.

10.3.4.3 sdh.dsa.cDSA.all_parts

A list of all the tactile sensor part indices of a finger of the SDH.

10.3.4.4 sdh.dsa.cDSA.calib_pressure

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

10.3.4.5 sdh.dsa.cDSA.calib_voltage

see calib_pressure

10.3.4.6 sdh.dsa.cDSA.com

10.3.4.7 sdh.dsa.cDSA.contact_area_cell_threshold

threshold of texel cell value for detecting contacts with GetContactArea

10.3.4.8 sdh.dsa.cDSA.contact_force_cell_threshold

threshold of texel cell value for detecting forces with GetContactForce

10.3.4.9 sdh.dsa.cDSA.controller_info

A structure holding info about the remote DSACON32m controller.

10.3.4.10 tuple sdh.dsa.cDSA.eDSAPacketID [static]

Initial value:

```
dict( eDSA_FULL_FRAME = 0x00,
                         eDSA_QUERY_CONTROLLER_CONFIGURATION = 0x01,
                         eDSA_QUERY_SENSOR_CONFIGURATION = 0x02,
                         eDSA_QUERY_MATRIX_CONFIGURATION = 0x0B,
                         eDSA_CONFIGURE_DATA_ACQUISITION = 0x03,
                         eDSA_QUERY_CONTROLLER_FEATURES = 0x10,
                         eDSA_READ_MATRIX_MASK = 0x04,
                         eDSA_SET_DYNAMIC_MASK = 0xAB,
                         eDSA_READ_DESCRIPTOR_STRING = 0x05,
                         eDSA_LOOP = 0x06,
                         eDSA_QUERY_CONTROLLER_STATE = 0x0a,
                         eDSA_SET_PROPERTIES_SAMPLE_RATE = 0x0c,
                         eDSA_SET_PROPERTIES_CONTROL_VECTOR_FOR_MATRIX = 0x0d,
                         eDSA_GET_PROPERTIES_CONTROL_VECTOR_OF_MATRIX = 0x0e,
                         eDSA_ADJUST_MATRIX_SENSITIVITY = 0x0f,
                         eDSA_GET_SENSITIVITY_ADJUSTMENT_INFO = 0x12,
                         eDSA_SET_MATRIX_THRESHOLD = 0x13,
                         eDSA_GET_MATRIX_THRESHOLD = 0x14
                        )
```

A dictionary of packet ID to number mappings.

10.3.4.11 list sdh.dsa.cDSA.error_codes [static]

Initial value:

```
[
              (0, "E_SUCCESS", "No error occurred, operation was successful
      "),
              (1, "E_NOT_AVAILABLE", "Function or data is not available"),
              (2, "E_NO_SENSOR", "No measurement converter is connected"),
              (3, "E_NOT_INITIALIZED", "Device was not initialized"),
              (4, "E_ALREADY_RUNNING", "The data acquisition is already run
      ning"),
              (5, "E_FEATURE_NOT_SUPPORTED", "The requested feature is curr
      ently not available"),
              (6, "E_INCONSISTENT_DATA", "One or more parameters are incons
      istent"),
              (7, "E_TIMEOUT", "Timeout error"),
              (8, "E_READ_ERROR", "Error while reading data"),
              (9, "E_WRITE_ERROR", "Error while writing data"),
              (10, "E_INSUFFICIENT_RESOURCES", "No more memory available"),

              (11, "E_CHECKSUM_ERROR", "Checksum error"),
              (12, "E_CMD_NOT_ENOUGH_PARAMS", "Not enough parameters for ex
      ecuting the command"),
              (13, "E_CMD_UNKNOWN", "Unknown command"),
              (14, "E_CMD_FORMAT_ERROR", "Command format error"),
              (15, "E_ACCESS_DENIED", "Access denied"),
              (16, "E_ALREADY_OPEN", "Interface is already open"),
              (17, "E_CMD_FAILED", "Error while executing a command"),
              (18, "E_CMD_ABORTED", "Command execution was aborted by the u
      ser"),
              (19, "E_INVALID_HANDLE", "Invalid handle"),
              (20, "E_DEVICE_NOT_FOUND", "Device not found"),
```

```
                (21, "E_DEVICE_NOT_OPENED", "Device not opened"),
                (22, "E_IO_ERROR", "Input/Output Error"),
                (23, "E_INVALID_PARAMETER", "Wrong parameter"),
                (24, "E_INDEX_OUT_OF_BOUNDS", "Index out of bounds"),
                (25, "E_CMD_PENDING", "No error, but the command was not comp
leted, yet. Another return message will follow including an error code, if the fu
nction was completed."),
                (26, "E_OVERRUN", "Data overrun"),
                (27, "E_RANGE_ERROR", "Range error")
                ]
```

A list of triples (error code (int), error code name (string), error code description (string)) These are the error codes reported by the remote DSACON32m controller.

These are contained as member `response.error_code` in the response structs returned by some member functions. The cDSAError exception that is thrown in case of an error contains this value as member error_code as well, if the error was reported by the remote DSACON32m controller.

10.3.4.12 sdh.dsa.cDSA.force_factor

additional calibration factor for forces in GetContactForce

10.3.4.13 sdh.dsa.cDSA.frame

A structure containing the last tactile sensor frame read from the SDH.

10.3.4.14 sdh.dsa.cDSA.GetTimeout

10.3.4.15 sdh.dsa.cDSA.matrix_info

A list of structures holding info about the remote tactile sensors connected to the DSACON32m.

10.3.4.16 sdh.dsa.cDSA.port

10.3.4.17 sdh.dsa.cDSA.read_another

flag, if True then the ReadFrame() function will read tactile sensor frames until a timeout occurs.

This will ignore intermediate frames as long as new ones are available. This was usefull some time in the past, or if you have a slow computer that cannot handle incoming frames fast enough. If False then any completely read valid frame will be reported. With new computers and fast communication like via TCP this should remain at "False".

10.3.4.18 sdh.dsa.cDSA.sensor_info

A structure holding info about the remote DSACON32m sensor.

10.3.4.19 sdh.dsa.cDSA.SetTimeout

10.3.4.20 sdh.dsa.cDSA.texel_offset

A list of texel offsets.

For each sensor matrix the offset of the first texel of the matrix in the frame is stored.

10.3.4.21 sdh.dsa.cDSA.timeout

The read timeout for accessing the interface in seconds.

This is a property of the superclass serial.Serial.

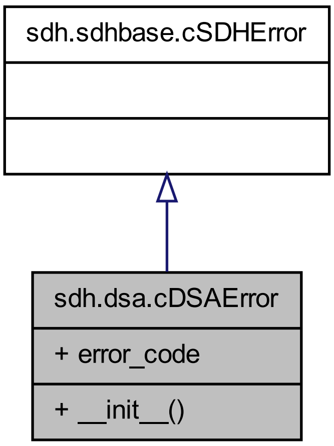
The documentation for this class was generated from the following file:

- sdh/dsa.py

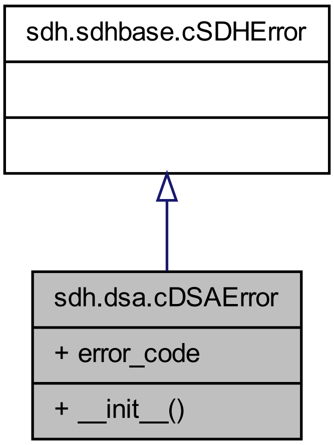
10.4 sdh.dsa.cDSAError Class Reference

DSA (tactile sensor of the SDH) related error occurred.

Inheritance diagram for sdh.dsa.cDSAError:



Collaboration diagram for sdh.dsa.cDSAError:



Public Member Functions

- def __init__

  Constructor for cDSAError exceptions.

Public Attributes

- error_code

10.4.1 Detailed Description

DSA (tactile sensor of the SDH) related error occurred.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 def sdh.dsa.cDSAError.__init__ (*self*, *msg*, *error_code* = `-1`)

Constructor for cDSAError exceptions.

The parameteter error_code is stored as a member. A positive value indicates an error code received from the remote DSACON32m controller, see cDSA.error_codes. A negative value indicates (library) internal errors.

10.4.3 Member Data Documentation

10.4.3.1 sdh.dsa.cDSAError.error_code

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

- sdh/dsa.py

10.5 demo-dsa.cMovingAverage Class Reference

Some additional classes and functions.

Public Member Functions

- def __init__

  Constructor, create a cMovingAverage object with the given window_size.

- def Reset

  Reset the internal state.

- def Add

  Add value v to the moving average calculation.

- def Get

  Calculate and return the current moving average.

Public Attributes

- window_size
- data
- next

10.5.1 Detailed Description

Some additional classes and functions. Class to implement objects that calculate a moving average

10.5.2 Constructor & Destructor Documentation

10.5.2.1 def demo-dsa.cMovingAverage.__init__ (*self*, *window_size* = 7)

Constructor, create a cMovingAverage object with the given window_size.

10.5.3 Member Function Documentation

10.5.3.1 def demo-dsa.cMovingAverage.Add (*self*, *v*)

Add value v to the moving average calculation.

10.5.3.2 def demo-dsa.cMovingAverage.Get (*self*)

Calculate and return the current moving average.

10.5.3.3 def demo-dsa.cMovingAverage.Reset (*self*, *window_size* = 7)

Reset the internal state.

10.5.4 Member Data Documentation

10.5.4.1 demo-dsa.cMovingAverage.data

10.5.4.2 demo-dsa.cMovingAverage.next

10.5.4.3 demo-dsa.cMovingAverage.window_size

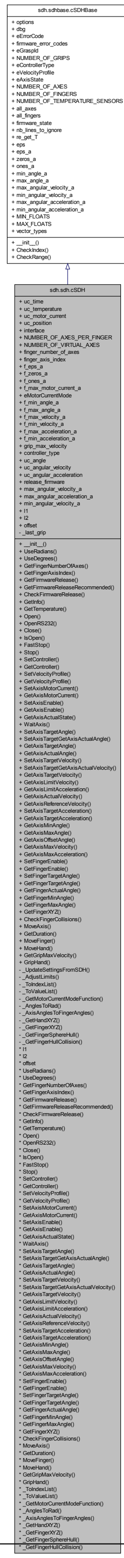
The documentation for this class was generated from the following file:

- demo/demo-dsa.py

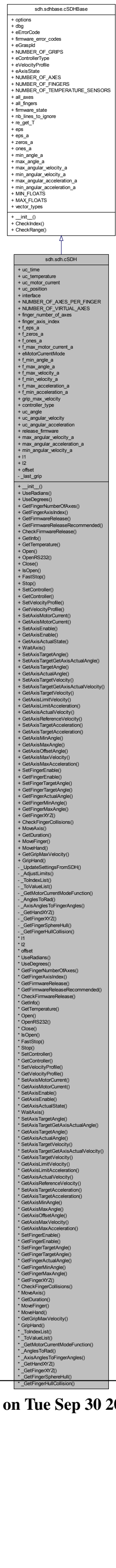
10.6 sdh.sdh.cSDH Class Reference

The actual SDH classes.

Inheritance diagram for sdh.sdh.cSDH:



Collaboration diagram for sdh.sdh.cSDH:



Public Member Functions

- def __init__

  Constructor of cSDH class.

Miscellaneous methods

- def UseRadians

  Shortcut to set the unit system to radians.

- def UseDegrees

  Shortcut to set the unit system to degrees.

- def GetFingerNumberOfAxes

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

- def GetFingerAxisIndex

  Return axis index of iFingerAxis axis of finger with index iFinger.

- def GetFirmwareRelease

  Return the actual release of the firmware of the SDH (not the library) as string.

- def GetFirmwareReleaseRecommended

  Return the recommended release of the firmware of the SDH by this library as string.

- def CheckFirmwareRelease

  Check the actual release of the firmware of the connected SDH against the recommended firmware release.

- def GetInfo

  Return info according to what.

- def GetTemperature

  Return temperature(s) measured within the SDH.

Communication methods

- def Open

  Open connection to SDH according to options.

- def OpenRS232

  Alias for Open() function for compatibility reasons.

- def Close

  Close connection to SDH.

- def IsOpen

  return wether the communication to the sdh is open or not

Auxiliary movement methods

- def FastStop
  Stop movement of all axes of the SDH and switch off the controllers.
- def Stop
  Stop movement of all axes but keep controllers on.
- def SetController
  Set the type of axis controller to be used in the SDH.
- def GetController
  Get the type of axis controller used in the SDH.
- def SetVelocityProfile
  Set the type of velocity profile to be used in the SDH.
- def GetVelocityProfile
  Get the type of velocity profile used in the SDH.

Methods to access SDH on axis-level

- def SetAxisMotorCurrent
  Set the maximum allowed motor current(s) for axis(axes).
- def GetAxisMotorCurrent
  Get the maximum allowed motor current(s) of axis(axes).
- def SetAxisEnable
  Set enabled/disabled state of axis controller(s).
- def GetAxisEnable
  Get enabled/disabled state of axis controller(s).
- def GetAxisActualState
  Get the current actual state(s) of axis(axes).
- def WaitAxis
  Wait until the movement(s) of of axis(axes) has finished.
- def SetAxisTargetAngle
  Set the target angle(s) for axis(axes).
- def SetAxisTargetGetAxisActualAngle
  Set the target angle(s) and read the actual angle(s) for axis(axes).
- def GetAxisTargetAngle
  Get the target angle(s) of axis(axes).

- def GetAxisActualAngle
  Get the current actual angle(s) of axis(axes).
- def SetAxisTargetVelocity
  Set the target velocity(s) for axis(axes).
- def SetAxisTargetGetAxisActualVelocity
  Set the target velocity(s) and get actual velocity(s) of axis(axes).
- def GetAxisTargetVelocity
  Get the target velocity(s) of axis(axes).
- def GetAxisLimitVelocity
  Get the velocity limit(s) of axis(axes).
- def GetAxisLimitAcceleration
  Get the acceleration limit(s) of axis(axes).
- def GetAxisActualVelocity
  Get the actual velocity(s) of axis(axes).
- def GetAxisReferenceVelocity
  Get the current reference velocity(s) of axis(axes).
- def SetAxisTargetAcceleration
  Set the target acceleration(s) for axis(axes).
- def GetAxisTargetAcceleration
  Get the target acceleration(s) of axis(axes).
- def GetAxisMinAngle
  Get the minimum angle(s) of axis(axes).
- def GetAxisMaxAngle
  Get the maximum angle(s) of axis(axes).
- def GetAxisOffsetAngle
  Get the offset angle(s) of axis(axes).
- def GetAxisMaxVelocity
  Get the maximum velocity(s) of axis(axes).
- def GetAxisMaxAcceleration
  Get the maximum acceleration(s) of axis(axes).

Methods to access SDH on finger-level

- def SetFingerEnable
  Set enabled/disabled state of axis controllers of finger(s).

- def GetFingerEnable
  Get enabled/disabled state of axis controllers of finger(s).
- def SetFingerTargetAngle
  Set the target angle(s) for a single finger.
- def GetFingerTargetAngle
  Get the target axis angles of a single finger.
- def GetFingerActualAngle
  Get the current actual axis angles of a single finger.
- def GetFingerMinAngle
  Get the minimum axis angles of a single finger.
- def GetFingerMaxAngle
  Get the maximum axis angles of a single finger.
- def GetFingerXYZ
  Get the xyz finger tip position of a single finger.
- def CheckFingerCollisions
  Check for internal collisions at the given finger angles.
- def MoveAxis
  Move one or more axes to the previously set target position with the previously set (maximum) velocities.
- def GetDuration
  Get Duration of movement of one or more axes to the previously set target position with the previously set (maximum) velocities.
- def MoveFinger
  Move a single finger to the previously set target position with the previously set (maximum) velocities.
- def MoveHand
  Move all selected fingers to the previously set target position with the previously set (maximum) velocities.

Methods to access SDH grip skills

- def GetGripMaxVelocity
  Get the maximum velocity of grip skills.
- def GripHand
  Perform one of the internal skills (a "grip").

Public Attributes

- uc_time

  unit convert for times: default = unit.uc_time_seconds

- uc_temperature

  unit convert for temperatures: default = unit.uc_temperature_celsius

- uc_motor_current

  unit converter for motor curent: default = unit.uc_motor_current_ampere

- uc_position

  unit converter for position: default = unit.uc_position_millimeter

- interface

  The interface to the SDH hardware:

- NUMBER_OF_AXES_PER_FINGER

  The number of axis per finger (for finger 1 this includes the "virtual" base axis)

- NUMBER_OF_VIRTUAL_AXES

  The number of virtual axes.

- finger_number_of_axes

  Mapping of finger index to number of real axes of fingers:

- finger_axis_index

  Mapping of finger index, finger axis index to axis index:

- f_eps_a

  array of 3 epsilon values

- f_zeros_a

  array of 3 0 values

- f_ones_a

  array of 3 1 values

- f_max_motor_current_a

  Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.

- eMotorCurrentMode

  the motor current can be set specifically for these modes:

- f_min_angle_a

Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.

- f_max_angle_a

  Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.

- f_max_velocity_a

  Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis we cannot read the real limits from the SDH firmware yet, since we are not yet connected.

- f_min_velocity_a

  Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis we cannot read the real limits from the SDH firmware yet, since we are not yet connected.

- f_max_acceleration_a

  Maximum allowed axis acceleration (in internal units (degrees/second²)), including the virtual axis.

- f_min_acceleration_a

  Minimum allowed axis acceleration (in internal units (degrees/second²)), including the virtual axis.

- grip_max_velocity

  Maximum allowed grip velocity (in internal units (degrees/second))

- controller_type
- uc_angle

  unit convert for (axis) angles: default = unit.uc_angle_degrees

- uc_angular_velocity

  unit convert for (axis) angular velocities: default = unit.uc_angular_velocity_degrees_-per_second

- uc_angular_acceleration

  unit convert for (axis) angular accelerations: default = unit.uc_angular_acceleration_-degrees_per_second_squared

- release_firmware
- max_angular_velocity_a

  Maximum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

- max_angular_acceleration_a

  Maximum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

- min_angular_velocity_a

Minimum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

Kinematic parameters of the Hand

- l1

  length of limb 1 (proximal joint to distal joint) in mm

- l2

  length of limb 2 (distal joint to fingertip) in mm

- offset

  list of xyz-arrays for all fingers with offset from (0,0,0) of proximal joint in mm x, y, z

10.6.1 Detailed Description

The actual SDH classes. The end user interface class to control a SDH (SCHUNK Dexterous Hand).

A general overview of the structure and architecture 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 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 parmeters: The interface functions defined here make full use of pythons flexibility. I.E. for parameters of functions like axis indices or axis

angles not only single numerical values can be given, but also python-lists, -tuples or -arrays of values. This way the same interface function can address a single axis individually or multiple axes in one call, as required by the application. Such parameters are herein refered to as "vectors". The actual allowed types for such vectors is determined by cSDHBase.vector_types.

- Parameters for methods are checked for validity. In case an invalid parameter is given the method raises a cSDHErrorInvalidParameter exception.
- The underlying physical unit system of parameters that 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 Python import module only (USERAPP.py and sdh.py in the overview figure"). For now the SDH firmware knows only about its internal unit system.

The actual class to control the SCHUNK Dexterous Hand. See html/pdf documentation for details.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 def sdh.sdh.cSDH.__init__ (*self*, *options* = `None`)

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 Open() 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 *options*) are:

- `degrees` for (axis) angles
- `degrees per second` for (axis) angular velocities
- `seconds` for times
- `degrees celsius` for temperatures

Parameters

self	- reference to the object itself

options	- a collection of additional settings, like returned e.g. from cSDHOptionParser.parse_args()
	• Settings used by the base class cSDHBase:
	  – `"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
	• Settings used by the interface class cSDHSerial:
	  – `"port"` : if set, then it is used as the port numter or device name of the serial port to use. The default value port=0 refers to 'COM1' in Windows and to the corresponding '/dev/ttyS0' in Linux.
	  – `"timeout"` : the timeout to use:
	    * None : wait forever
	    * T : wait for T seconds (float accepted)
	• Settings used by this cSDH class only:
	  – `"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

Examples:

Common use:

```
# Import the sdh.py python import module:
import sdh

# Create a cSDH object 'hand'. This calls the constructor sdh.cSDH.__init__()
  :
hand = sdh.cSDH()
```

The mentioned change of a unit system can be done like this:

```
# Assuming 'hand' is a sdh.cSDH object ...

# override default unit converter for (axis) angles:
hand.uc_angle = sdh.uc_angle_radians

# override default unit converter for (axis) angular velocities:
hand.uc_angular_velocity = sdh.uc_angular_velocity_radians_per_second

# override default unit converter for (axis) angular accelerations:
hand.uc_angular_acceleration = sdh.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  = sdh.uc_time_milliseconds

# override default unit converter for temperatures:
hand.uc_temperature = sdh.uc_temperature_fahrenheit

# override default unit converter for positions:
hand.uc_position = sdh.uc_position_meter
```

For convenience the most common settings can be specified as a dictionary-style parameter *options* for the constructor.

This can be done either manually, like in:

```
#import the sdh.py python
import module import sdh

# Create a cSDH object 'hand'
hand = sdh.cSDH( options=dict( port=1, debug_level=2, use_radians=True) )
```

Or for the deluxe, 'all batteries included' variant, the options can be used automagically from a cSDHOptionParser object, like in:

```
#import the sdh.py python
import module import sdh

# Create an option parser object to parse common command line options:
parser = sdh.cSDHOptionParser( usage    = "Your mighty explanative descriptio
  n",
                               revision = "0.0-pre_alpha" )

# Parse (and handle, if possible) the command line options of the script:
(options, args) = parser.parse_args()

# Create a cSDH object 'hand' using the parsed options:
hand = sdh.cSDH( options=options.__dict__ )
# (We cannot use options directly, but its __dict__ attribut
# holds the dictionary that the constructor needs)
```

Constructor of cSDH class. See html/pdf documentation for details.

Reimplemented from sdh.sdhbase.cSDHBase.

10.6.3 Member Function Documentation

10.6.3.1 def sdh.sdh.cSDH.CheckFingerCollisions (*self*, *f0aa* = None, *f1aa* = None, *f2aa* = None, *iv_filename* = None)

Check for internal collisions at the given finger angles.

Parameters

self	- reference to the object itself
f0aa	- a vector of NUMBER_OF_AXES_PER_FINGER angles for finger 0. If the default None is used then the current actual axis angles are read from the SDH and used.

f1aa	- a vector of NUMBER_OF_AXES_PER_FINGER angles for finger 1. If the default `None` is used then the current actual axis angles are read from the SDH and used.
f2aa	- a vector of NUMBER_OF_AXES_PER_FINGER angles for finger 2. If the default `None` is used then the current actual axis angles are read from the SDH and used.
iv_filename	- A filename for an OpenInventor ivfile to generate or None

Returns

a tuple (cxy, (c01, d01), (c02,d02), (c12, d12)) with:

- cxy is True if there are any internal finger collisions
- c01 is True if finger 0 and 1 collide
- c02 is True if finger 0 and 2 collide
- c12 is True if finger 1 and 2 collide
- d01 is the minimum distance of fingers 0 and 1
- d02 is the minimum distance of fingers 0 and 2
- d12 is the minimum distance of fingers 1 and 2

Remarks

- The angle values are expected in the configured angle unit system uc_angle.
- the returned distance is given in the configured position unit system uc_-position.
- If all the angle values are given (not `None`) then no communication is performed with the SDH. This function can then be used 'offline'.
- The finger joint angles must be given as the corresponding parameter, i.e. giving the joint angles of finger 0 as f2aa and those of finger 2 as f0aa will NOT work!

10.6.3.2 def sdh.sdh.cSDH.CheckFirmwareRelease (*self*)

Check the actual release of the firmware of the connected SDH against the recommended firmware release.

Returns

true - if the actual firmware is the recommended one false - the actual firmware is NOT the recommended one (communication with the SDH might not work as expected)

This will throw a (cSDHErrorCommunication∗) exception if the connection to the SDH is not yet opened.

Examples:

```
// Assuming 'hand' is a cSDH object ...

if ( hand.CheckFirmwareRelease() )
{
    cout << "The firmware release of the connected SDH is the one recommende
 d by this SDHLibrary\n";
}
else
{
    cout << "The firmware release of the connected SDH is NOT the one recomm
 ended by this SDHLibrary\n";
    cout << "  Actual SDH firmware release      " << hand.GetFirmwareRelease
 () << "\n";
    cout << "  Recommended SDH firmware release " << hand.GetFirmwareRelease
 Recommended() << "\n";
}
```

See also

See GetFirmwareReleaseRecommended() to get the recommended SDH firmware release.

10.6.3.3 def sdh.sdh.cSDH.Close (*self*, *leave_enabled* = `False`)

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

self	- reference to the object itself
leave_-enabled	- Flag: True to leave the controllers on, False (default) to disable the controllers (switch powerless)

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Close connection to SDH, power off controllers:
hand.Close()

# To leave the already enabled controllers enabled:
hand.Close( True )
```

Close connection to SDH.

10.6.3.4 def sdh.sdh.cSDH.FastStop (*self*)

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 performed and confirmed the fast 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 sdh.cSDH object ...

# Perform an fast stop:
hand.FastStop()
```

Stop movement of all axes of the SDH and switch off the controllers. See html/pdf documentation for details.

10.6.3.5 def sdh.sdh.cSDH.GetAxisActualAngle (*self*, *iAxis* = `All`)

Get the current actual angle(s) of axis(axes).

The actual angles are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes actual angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the current actual angle of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get actual axis angle of axis 3
v = hand.GetAxisActualAngle( 3 )
# v is now something like 42.0
```

```
# Get actual axis angle of axis 3
L = hand.GetAxisActualAngle( [3] )
# now L is something like [42.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get actual axis angle of axis 3 and 5
L = hand.GetAxisActualAngle( [3,5] )
# now L is something like [42.0, 47.11]

# Get actual axis angle of all axes
L = hand.GetAxisActualAngle( sdh.All )
# now L is something like [0.0, 0.0, 42.0, 0.0, 47.11, 0,0, 0.0]

# Get actual axis angle of all axes
L = hand.GetAxisActualAngle()
# now L is something like [0.0, 0.0, 42.0, 0.0, 47.11, 0,0, 0.0]
```

Get the current actual angle(s) of axis(axes)

10.6.3.6 def sdh.sdh.cSDH.GetAxisActualState (*self*, *iAxis* = `All`)

Get the current actual state(s) of axis(axes).

The actual axis states are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single int value is returned
- else a list of the selected axes actual states is returned
- The value(s) are returned numerically, (see also cSDHBase.eAxisState):
  - 0 : controller on but not moving
  - 1 : controller on and moving
  - 6 : controller off

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the current actual state of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...
```

```
# Get actual axis state of axis 3
v = hand.GetAxisActualState( 3 )
# v is now something like 0

# Get actual axis state of axis 3
L = hand.GetAxisActualState( [3] )
# now L is something like [0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get actual axis state of axis 3 and 5
L = hand.GetAxisActualState( [3,5] )
# now L is something like [0, 6]

# Get actual axis state of all axes
L = hand.GetAxisActualState( sdh.All )
# now L is something like [0, 0, 0, 0, 0, 6, 0]

# Get actual axis state of all axes
L = hand.GetAxisActualState()
# now L is something like [0, 0, 0, 0, 0, 6, 0]
```

Get the current actual state(s) of axis(axes)

10.6.3.7 def sdh.sdh.cSDH.GetAxisActualVelocity (*self*, *iAxis* = `All`)

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

The actual velocities are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes actual velocities is returned
- The value(s) are reported in the configured angular velocity unit system uc_angular_velocity.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the actual velocity of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...
```

```
# Get actual axis velocity of axis 3
v = hand.GetAxisActualVelocity( 3 )
# v is now something like 13.2

# Get actual axis velocity of axis 3
L = hand.GetAxisActualVelocity( [3] )
# now L is something like [13.2]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get actual axis velocity of axis 3 and 5
L = hand.GetAxisActualVelocity( [3,5] )
# now L is something like [13.2, 0.0]

# Get actual axis velocity of all axes
L = hand.GetAxisActualVelocity( sdh.All )
# now L is something like [0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7]

# Get actual axis velocity of all axes
L = hand.GetAxisActualVelocity()
# now L is something like [0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7]
```

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

10.6.3.8 def sdh.sdh.cSDH.GetAxisEnable (*self*, *iAxis* = `All`)

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

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single int value (0\|1) is returned
- else a list of the selected axes enabled states as int values (0\|1) is returned

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the enabled state of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given).

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Get enabled state of all axes:
L = hand.GetAxisEnable( sdh.All )
# now L is something like [0,0,0,1,1,0,0]

# Get enabled state of axis 0 and 3
L = hand.GetAxisEnable( [0,3] )
# now L is something like [0,1]

# Get enabled state of axis 3
v = hand.GetAxisEnable( 3 )
# now v is something like 1

# Get enabled state of axis 2
L = hand.GetAxisEnable( [2] )
# now L is something like [0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)
```

Get enabled/disabled state of axis/axes. See html/pdf documentation for details.

10.6.3.9 def sdh.sdh.cSDH.GetAxisLimitAcceleration (*self*, *iAxis* = `All`)

Get the acceleration limit(s) of axis(axes).

The acceleration limit(s) are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes acceleration limits is returned
- The value(s) are reported in the configured angular acceleration unit system uc_angular_acceleration.

Remarks

- The order of the returned list depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the target acceleration of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get axis acceleration limit of axis 3
a = hand.GetAxisLimitAcceleration( 3 )
# a is now something like 400.0

# Get axis acceleration limit of axis 3
L = hand.GetAxisLimitAcceleration( [3] )
# now L is something like [400.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get axis acceleration limit of axis 3 and 5
L = hand.GetAxisLimitAcceleration( [3,5] )
# now L is something like [400.0, 400.0]

# Get axis acceleration limit of all axes
L = hand.GetAxisLimitAcceleration( sdh.All )
# now L is something like [5000.0, 400.0, 1500.0, 400.0, 1500.0, 400.0, 1500.
  0]

# Get axis acceleration limit of all axes
L = hand.GetAxisLimitAcceleration()
# now L is something like [5000.0, 400.0, 1500.0, 400.0, 1500.0, 400.0, 1500.
  0]
```

Get the acceleration limit(s) of axis(axes)

10.6.3.10 def sdh.sdh.cSDH.GetAxisLimitVelocity (*self*, *iAxis* = `All`)

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

The velocity limit(s) are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes velocity limits is returned
- The value(s) are reported in the configured angular velocity unit system uc_-angular_velocity.

Remarks

- The order of the returned list depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the target velocity of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get axis velocity limit of axis 3
v = hand.GetAxisLimitVelocity( 3 )
# v is now something like 14.0

# Get axis velocity limit of axis 3
L = hand.GetAxisLimitVelocity( [3] )
# now L is something like [140.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get axis velocity limit of axis 3 and 5
L = hand.GetAxisLimitVelocity( [3,5] )
# now L is something like [140.0, 140.0]

# Get axis velocity limit of all axes
L = hand.GetAxisLimitVelocity( sdh.All )
# now L is something like [81.0, 140.0, 120.0, 140.0, 120.0, 140.0, 120.0]

# Get axis velocity limit of all axes
L = hand.GetAxisLimitVelocity()
# now L is something like [81.0, 140.0, 120.0, 140.0, 120.0, 140.0, 120.0]
```

Get the velocity limit(s) of axis(axes)

10.6.3.11 def sdh.sdh.cSDH.GetAxisMaxAcceleration (*self*, *iAxis* = `All`)

Get the maximum acceleration(s) of axis(axes).

The maximum accelerations are currently not read from the SDH, but are stored in the base class.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes maximum accelerations is returned

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the maximum acceleration of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get maximum axis acceleration of axis 3
v = hand.GetAxisMaxAcceleration( 3 )
# v is now something like 1000.0

# Get maximum axis acceleration of axis 3
L = hand.GetAxisMaxAcceleration( [3] )
# now L is something like [1000.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get maximum axis acceleration of axis 3 and 5
L = hand.GetAxisMaxAcceleration( [3,5] )
# now L is something like [1000.0, 1000.0]

# Get maximum axis acceleration of all axes
L = hand.GetAxisMaxAcceleration( sdh.All )
# now L is something like [1000.0, 1000.0, 1000.0, 1000.0, 1000.0, 1000.0, 10
  00.0]

# Get maximum axis acceleration of all axes
L = hand.GetAxisMaxAcceleration()
# now L is something like [1000.0, 1000.0, 1000.0, 1000.0, 1000.0, 1000.0, 10
  00.0]

# Or if you change the acceleration unit system:
hand.UseRadians()
L = hand.GetAxisMaxAcceleration()
# now L is something like [17.4532925, 17.4532925, 17.4532925, 17.4532925, 17
  .4532925, 17.4532925, 17.4532925]
```

Get the maximum acceleration(s) of axis(axes)

10.6.3.12 def sdh.sdh.cSDH.GetAxisMaxAngle (*self*, *iAxis* = All)

Get the maximum angle(s) of axis(axes).

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes maximum angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the maximum angle of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get maximum axis angle of axis 3
v = hand.GetAxisMaxAngle( 3 )
# v is now something like 90.0

# Get maximum axis angle of axis 3
L = hand.GetAxisMaxAngle( [3] )
# now L is something like [90.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get maximum axis angle of axis 3 and 5
L = hand.GetAxisMaxAngle( [3,5] )
# now L is something like [90.0, 90.0]

# Get maximum axis angle of all axes
L = hand.GetAxisMaxAngle( sdh.All )
# now L is something like [90.0, 90.0, 90.0, 90.0, 90.0, 90.0, 90.0]

# Get maximum axis angle of all axes
L = hand.GetAxisMaxAngle()
# now L is something like [90.0, 90.0, 90.0, 90.0, 90.0, 90.0, 90.0]

# Or if you change the angle unit system:
hand.UseRadians()
L = hand.GetAxisMaxAngle()
# now L is something like [1.5707963267948966, 1.5707963267948966, 1.57079632
  67948966, 1.5707963267948966, 1.5707963267948966, 1.5707963267948966, 1.570796326
  7948966]
```

Get the maximum angle(s) of axis(axes)

10.6.3.13 def sdh.sdh.cSDH.GetAxisMaxVelocity (*self*, *iAxis* = `All`)

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

The maximum velocitys are currently not read from the SDH, but are stored in the base class.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes maximum velocitys is returned

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the maximum velocity of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get maximum axis velocity of axis 3
v = hand.GetAxisMaxVelocity( 3 )
# v is now something like 100.0

# Get maximum axis velocity of axis 3
L = hand.GetAxisMaxVelocity( [3] )
# now L is something like [100.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get maximum axis velocity of axis 3 and 5
L = hand.GetAxisMaxVelocity( [3,5] )
# now L is something like [100.0, 100.0]

# Get maximum axis velocity of all axes
L = hand.GetAxisMaxVelocity( sdh.All )
# now L is something like [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0]

# Get maximum axis velocity of all axes
L = hand.GetAxisMaxVelocity()
# now L is something like [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0]

# Or if you change the velocity unit system:
hand.UseRadians()
L = hand.GetAxisMaxVelocity()
# now L is something like [0.488692190, 1.74532925, 1.74532925, 1.74532925, 1
  .74532925, 1.74532925, 1.74532925]
```

Get the maximum velocity(s) of axis(axes)

10.6.3.14 def sdh.sdh.cSDH.GetAxisMinAngle (*self*, *iAxis* = `All`)

Get the minimum angle(s) of axis(axes).

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes minimum angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the minimum angle of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get minimum axis angle of axis 3
v = hand.GetAxisMinAngle( 3 )
# v is now something like -90.0

# Get minimum axis angle of axis 3
L = hand.GetAxisMinAngle( [3] )
# now L is something like [-90.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get minimum axis angle of axis 3 and 5
L = hand.GetAxisMinAngle( [3,5] )
# now L is something like [-90.0, -90.0]

# Get minimum axis angle of all axes
L = hand.GetAxisMinAngle( sdh.All )
# now L is something like [0.0, -90.0, -90.0, -90.0, -90.0, -90.0, -90.0]

# Get minimum axis angle of all axes
L = hand.GetAxisMinAngle()
# now L is something like [0.0, -90.0, -90.0, -90.0, -90.0, -90.0, -90.0]

# Or if you change the angle unit system:
hand.UseRadians()
L = hand.GetAxisMinAngle()
```

```
# now L is something like [0.0, -1.5707963267948966, -1.5707963267948966, -1.
  5707963267948966, -1.5707963267948966, -1.5707963267948966, -1.5707963267948966]
```

Get the minimum angle(s) of axis(axes)

10.6.3.15 def sdh.sdh.cSDH.GetAxisMotorCurrent (*self*, *iAxis* = `All`, *mode* = `0`)

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

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
mode	- move (default): The motor currents used while "moving" with a MoveHand() or MoveFinger() command • grip : The motor currents used while "gripping" with a GripHand() command • hold : The motor currents used after "gripping" with a GripHand() command (i.e. "holding")

Get the maximum allowed motor current(s) of axis(axes)

10.6.3.16 def sdh.sdh.cSDH.GetAxisOffsetAngle (*self*, *iAxis* = `All`)

Get the offset angle(s) of axis(axes).

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes offset angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the offset angle of axis *iAxis*[i] (not axis `i`).

- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get offset axis angle of axis 3
v = hand.GetAxisMaxAngle( 3 )
# v is now something like 3.45

# Get offset axis angle of axis 3
L = hand.GetAxisMaxAngle( [3] )
# now L is something like [3.45]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get offset axis angle of axis 3 and 5
L = hand.GetAxisMaxAngle( [3,5] )
# now L is something like [3.45, 5.67]

# Get offset axis angle of all axes
L = hand.GetAxisMaxAngle( sdh.All )
# now L is something like [0.12, 1.23, 2.34, 3.45, 4.56, 5.67, 6.78]

# Get offset axis angle of all axes
L = hand.GetAxisMaxAngle()
# now L is something like [0.12, 1.23, 2.34, 3.45, 4.56, 5.67, 6.78]

# Or if you change the angle unit system:
hand.UseRadians()
L = hand.GetAxisMaxAngle()
# now L is something like [0.0020943951023931952, 0.021467549799530253, 0.040
  84070449666731, 0.06021385919380437, 0.079587013890941416, 0.09896016858807849, 0
  .11833332328521554]
```

Get the offset angle(s) of axis(axes)

10.6.3.17 def sdh.sdh.cSDH.GetAxisReferenceVelocity (*self*, *iAxis* = `All`)

Get the current reference velocity(s) of axis(axes).

(This velocity is used internally by the SDH in eCT_VELOCITY_ACCELERATION mode)

The reference velocities are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes reference velocities is returned
- The value(s) are reported in the configured angular velocity unit system uc_-angular_velocity.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the reference velocity of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given
- 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.
- The availability of an appropriate SDH firmware is **not** checked here due to performance losses when this function is used often.

Examples:

```
# Assuming "hand" is a sdh.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 axis 3
v = hand.GetAxisReferenceVelocity( 3 )
# v is now something like 13.2

# Get reference axis velocity of axis 3
L = hand.GetAxisReferenceVelocity( [3] )
# now L is something like [13.2]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get reference axis velocity of axis 3 and 5
L = hand.GetAxisReferenceVelocity( [3,5] )
# now L is something like [13.2, 0.0]

# Get reference axis velocity of all axes
L = hand.GetAxisReferenceVelocity( sdh.All )
# now L is something like [0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7]

# Get reference axis velocity of all axes
L = hand.GetAxisReferenceVelocity()
# now L is something like [0.1, 0.2, 0.3, 13.2, 0.5, 0.0, 0.7]
```

Get the reference velocity(s) of axis(axes)

10.6.3.18 def sdh.sdh.cSDH.GetAxisTargetAcceleration (*self*, *iAxis* = `All`)

Get the target acceleration(s) of axis(axes).

The target accelerations are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes target accelerations is returned
- The value(s) are reported in the configured angular acceleration unit system uc_angular_acceleration.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the target acceleration of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get target axis acceleration of axis 3
v = hand.GetAxisTargetAcceleration( 3 )
# v is now something like 100.0

# Get target axis acceleration of axis 3
L = hand.GetAxisTargetAcceleration( [3] )
# now L is something like [100.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get target axis acceleration of axis 3 and 5
L = hand.GetAxisTargetAcceleration( [3,5] )
# now L is something like [100.0, 100.0]

# Get target axis acceleration of all axes
L = hand.GetAxisTargetAcceleration( sdh.All )
# now L is something like [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0 ]

# Get target axis acceleration of all axes
L = hand.GetAxisTargetAcceleration()
# now L is something like [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0 ]
```

Get the target acceleration(s) of axis(axes)

10.6.3.19 def sdh.sdh.cSDH.GetAxisTargetAngle (*self*, *iAxis* = `All`)

Get the target angle(s) of axis(axes).

The target angles are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes target angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the target angle of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given).

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get target axis angle of axis 3
v = hand.GetAxisTargetAngle( 3 )
# v is now something like 42.0

# Get target axis angle of axis 3
L = hand.GetAxisTargetAngle( [3] )
# now L is something like [42.0]
#    (if a vector is given as parameter then a list is returned,
#     even if it contains a single element only)

# Get target axis angle of axis 3 and 5
L = hand.GetAxisTargetAngle( [3,5] )
# now L is something like [42.0, 47.11]

# Get target axis angle of all axes
L = hand.GetAxisTargetAngle( sdh.All )
# now L is something like [0.0, 0.0, 42.0, 0.0, 47.11, 0,0, 0.0]

# Get target axis angle of all axes
L = hand.GetAxisTargetAngle()
# now L is something like [0.0, 0.0, 42.0, 0.0, 47.11, 0,0, 0.0]
```

Get the target angle(s) of axis(axes)

10.6.3.20 def sdh.sdh.cSDH.GetAxisTargetVelocity (*self*, *iAxis* = `All`)

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

The target velocities are read from the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes target velocities is returned
- The value(s) are reported in the configured angular velocity unit system uc_-angular_velocity.

Remarks

- The order of the returned list (if any) depends on the order of the axis indices given in *iAxis*. I.E. if a list `rc` is returned, then `rc`[i] will be the target velocity of axis *iAxis*[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get target axis velocity of axis 3
v = hand.GetAxisTargetVelocity( 3 )
# v is now something like 14.0

# Get target axis velocity of axis 3
L = hand.GetAxisTargetVelocity( [3] )
# now L is something like [14.0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)

# Get target axis velocity of axis 3 and 5
L = hand.GetAxisTargetVelocity( [3,5] )
# now L is something like [14.0, 12.34]

# Get target axis velocity of all axes
L = hand.GetAxisTargetVelocity( sdh.All )
# now L is something like [100.0, 40.0, 40.0, 14.0, 40.0, 12.34, 40.0]

# Get target axis velocity of all axes
L = hand.GetAxisTargetVelocity()
# now L is something like [100.0, 40.0, 40.0, 14.0, 40.0, 12.34, 40.0]
```

Get the target velocity(s) of axis(axes)

10.6.3.21 def sdh.sdh.cSDH.GetController (*self*)

Get the type of axis controller used in the SDH.

The currently set controller type will be queried and returned (One of self.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
print "Currently the axis controller type is set to %d (%s)" % (ct, [ v for (
  k,v) in hand.eControllerType.iteritems() if v == ct ][0])
```

Get the type of axis controller used in the SDH. See html/pdf documentation for details.

10.6.3.22 def sdh.sdh.cSDH.GetDuration (*self, iAxis*)

Get Duration of movement of one or more axes to the previously set target position with the previously set (maximum) velocities.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.

Returns

The expected 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.
- Other axes than *iAxis* will **NOT** move, even if target axis angles for their axes have been set.
- If *sequ* is True then the currently set target axis angles for other axes will be restored upon return of the function.
- If *sequ* is False then the currently set target axis angles for other axes will be overwritten with their current actual axis angles

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set a new target pose for axis 0:
hand.SetAxisTargetAngle( 0, 0 )

# Set a new target pose for axis 1,2 and 3
hand.SetAxisTargetAngle( [1, 2, 3], [-10,-20,-30] )
```

```
# Get duration for moving axis 0 only
t = hand.GetDuration( 0 )
# The axis 0 would move for t seconds.

# The target poses for axis 1,2 and 3 are still set since the
# last GetDuration() call did not really change the target positions set.
# So get duration of movement of axis 1 and 2 now:
t = hand.GetDuration( [1,2] )
```

Get duration of a movement of an axis or some axes to the previously set target positions with the previously set (maximum) target velocities. See html/pdf documentation for details.

10.6.3.23 def sdh.sdh.cSDH.GetFingerActualAngle (*self, iFinger*)

Get the current actual axis angles of a single finger.

The current actual axis angles of finger *iFinger* are read from the SDH.

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index.

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.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get actual axis angles of finger 0
L = hand.GetFingerActualAngle( 0 )
# L is now something like [42.0, -10.0, 47.11]

# Get actual axis angles of finger 1
L = hand.GetFingerActualAngle( 1 )
# now L is something like [0.0, 24.7, -5.5]
```

Get the current actual axis angles of a single finger. See html/pdf documentation for details.

10.6.3.24 def sdh.sdh.cSDH.GetFingerAxisIndex (*self, iFinger, iFingerAxis*)

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

self	- reference to the object itself
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*

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

print "The 1st axis of finger 2 has real axis index %d" % ( hand.GetFingerNum
  berOfAxes( 2, 0 ) )
```

Return axis index of iFingerAxis axis of finger with index iFinger

10.6.3.25 def sdh.sdh.cSDH.GetFingerEnable (*self*, *iFinger* = `All`)

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

self	- reference to the object itself
iFinger	- index of finger to access. This can be All, a single index or a vector of indices.

Returns

- if *iFinger* is a single index then a single int value (0|1) is returned
- else a list of the selected fingers enabled states as int values (0|1) is returned

Remarks

- The order of the returned list (if any) depends on the order of the finger indices given in *iFinger*. I.E. if a list `rc` is returned, then `rc`[i] will be the enabled state of finger *iFinger*[i] (not finger `i`).
- If *iFinger* is `All` then the order will be the natural one (as if *iFinger*=[0,1,2] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get enabled state of all fingers:
L = hand.GetFingerEnable( sdh.All )
# now L is something like [0,1,0]
```

```
# Get enabled state of finger 0 and 2
L = hand.GetFingerEnable( [0,2] )
# now L is something like [0,0]

# Get enabled state of finger 1
v = hand.GetFingerEnable( 1 )
# now v is something like 1

# Get enabled state of finger 0
L = hand.GetFingerEnable( [0] )
# now L is something like [0]
#   (if a vector is given as parameter then a list is returned,
#    even if it contains a single element only)
```

Get enabled/disabled state of finger(s). See html/pdf documentation for details.

10.6.3.26 def sdh.sdh.cSDH.GetFingerMaxAngle (*self*, *iFinger*)

Get the maximum axis angles of a single finger.

The maximum axis angles of finger *iFinger* are read from the base class and returned.

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index

Returns

- A list of the selected fingers maximum axis angles
- The values are returned in the configured angle unit system uc_angle.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get maximum axis angles of finger 0
L = hand.GetFingerMaxAngle( 0 )
# now L is something like [90.0, 90.0, 90.0]

# Get maximum axis angles of finger 1
L = hand.GetFingerMaxAngle( 1 )
# now L is something like [0.0, 90.0, 90.0]

# Or if you change the angle unit system:
hand.UseRadians()
L = hand.GetFingerMaxAngle( 0 )
# now L is something like [1.5707963267948966, 1.5707963267948966, 1.57079632
  67948966]
```

Get the maximum axis angles of a single finger. See html/pdf documentation for details.

10.6.3.27 def sdh.sdh.cSDH.GetFingerMinAngle (*self*, *iFinger*)

Get the minimum axis angles of a single finger.

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index

Returns

- A list of the selected fingers minimum axis angles
- The values are returned in the configured angle unit system uc_angle.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get minimum axis angles of finger 0
L = hand.GetFingerMinAngle( 0 )
# now L is something like [0.0, -90.0, -90.0]

# Get minimum axis angles of finger 1
L = hand.GetFingerMinAngle( 1 )
# now L is something like [0.0, -90.0, -90.0]

# Or if you change the angle unit system:
hand.UseRadians()
L = hand.GetFingerMinAngle( 0 )
# now L is something like [0.0, -1.5707963267948966, -1.5707963267948966]
```

Get the minimum axis angles of a single finger. See html/pdf documentation for details.

10.6.3.28 def sdh.sdh.cSDH.GetFingerNumberOfAxes (*self*, *iFinger*)

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

Parameters

self	- reference to the object itself
iFinger	- index of finger in range [0..NUMBER_OF_FINGERS-1]

Returns

number of real axes of finger with index *iFinger*

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

print "The finger 0 has %d real axes" % ( hand.GetFingerNumberOfAxes( 0 ) )
```

Return the number of axes of finger with index iFinger.

10.6.3.29 def sdh.sdh.cSDH.GetFingerTargetAngle (*self*, *iFinger*)

Get the target axis angles of a single finger.

The target axis angles of finger *iFinger* are read from the SDH.

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index

Returns

- A list of the selected fingers target axis angles
- The values are returned in the configured angle unit system uc_angle.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get target axis angle of finger 0
L = hand.GetFingerTargetAngle( 0 )
# now L is something like [42.0, -10.0, 47.11]

# Get target axis angle of finger 1
L = hand.GetFingerTargetAngle( 1 )
# now L is something like [0.0, 24.7, -5.5]
```

Get the target axis angles of a single finger. See html/pdf documentation for details.

10.6.3.30 def sdh.sdh.cSDH.GetFingerXYZ (*self*, *iFinger*, *angles* = `None`)

Get the xyz finger tip position of a single finger.

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index
angles	- a vector of NUMBER_OF_AXES_PER_FINGER angles (in external units, see uc_angle) If the default `None` is used then the current actual axis angles are read from the SDH and used. The values are expected in the configured angle unit system uc_angle.

Returns

- A list of the x,y,z values of the finger tip position
- The values are returned in the configured position unit system uc_position.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get actual finger tip position of finger 0:
P = hand.GetFingerXYZ( 0 )
# now P is something like [18.821618775581801, 32.600000000000001, 174.0]
# if finger 0 is at axis angles [0,0,0]
```

```
# Get finger tip position of finger 0 at axis angles [90,-90,-90]:
P = hand.GetFingerXYZ( 0, [90,-90,-90] )
# now P is something like [18.821618775581804, 119.60000000000002, -53.0]

# Or if you change the angle unit system:
hand.UseRadians()
P = hand.GetFingerXYZ( 0, [1.5707963267948966, -1.5707963267948966, -1.570796
  3267948966] )
# now P is still something like [18.821618775581804, 119.60000000000002, -53.
  0]

# Or if you change the position unit system too:
hand.uc_position = sdh.uc_position_meter
P = hand.GetFingerXYZ( 0, [1.5707963267948966, -1.5707963267948966, -1.570796
  3267948966] )
# now P is still something like [0.018821618775581, 0.119.60000000000002, -0.
  052999999999]
```

Get the xyz finger tip position of a single finger.

10.6.3.31 def sdh.sdh.cSDH.GetFirmwareRelease (*self*)

Return the actual release 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 sdh.cSDH object ...

print "The SDH firmware reports release ", hand.GetFirmwareRelease()
```

See also

See GetFirmwareReleaseRecommended() to get the actual release of the SDH firmware

10.6.3.32 def sdh.sdh.cSDH.GetFirmwareReleaseRecommended ( )

Return the recommended release of the firmware of the SDH by this library as string.

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

print "This SDHLibrary recommends an SDH firmware release", hand.GetFirmware
 ReleaseRecommended()
```

See also

See GetFirmwareRelease() to get the actual release of the SDH firmware

10.6.3.33 def sdh.sdh.cSDH.GetGripMaxVelocity (*self*)

Get the maximum velocity of grip skills.

The maximum velocity is currently not read from the SDH, but is stored in the base class.

Returns

- a single float value is returned representing the velocity in the uc_angular_velocity unit system

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Get maximum grip skill velocity
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
```

Get the maximum velocity of axis(axes)

10.6.3.34 def sdh.sdh.cSDH.GetInfo (*self*, *what*)

Return info according to *what*.

The following values are valid for *what:*

- "date-library" : date of the SDHLibrary-python release
- "release-library" : release name of the sdh.py python module
- "release-firmware" : release name of the SDH firmware (requires an opened communication to the SDH)
- "release-firmware-recommended" : recommended release name of the SDH firmware
- "date-firmware" : date of the SDH firmware (requires an opened communication to the SDH)
- "release-soc" : release name of the SDH SoC (requires an opened communication to the SDH)
- "date-soc" : date of the SDH SoC (requires an opened communication to the SDH)
- "id-sdh" : ID of SDH
- "sn-sdh" : Serial number of SDH

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

print "The SDH firmware reports release %s" % ( hand.GetInfo( "release-firmwa
  re" ) )
```

Return info according to *what*. See html/pdf documentation for details.

10.6.3.35 def sdh.sdh.cSDH.GetTemperature (*self*, *iSensor* = All)

Return temperature(s) measured within the SDH.

Parameters

self	- reference to the object itself
iSensor	- index of temperature sensor to access. This can be All, a single index or a vector of indices. • 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)

Returns

The temperature(s) returned are reported in the configured temperature unit system uc_temperature.

- if iSensor is a single index then a single float value is returned
- else a list of the selected sensor values is returned

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Get measured values of all sensors
temps = hand.GetTemperature()
# 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:
temp_controller = hand.GetTemperature( 0 )
# Now temp_controller is something like 38.5

# If we - for some obscure islandish reason - would want
# temperatures reported in degrees fahrenheit, the unit
# converter can be changed:
```

```
hand.uc_temperature = sdh.uc_temperature_fahrenheit

# Get all temperaturs again:
temps_f = hand.GetTemperature()
# Now temps_f is something like [ 100.0, 96.8, 92.3, 97.7, 91.8, 96.8, 90.1,
   137.5,  125.2]
```

Return temperature(s) measured within the SDH. See html/pdf documentation for details.

10.6.3.36 def sdh.sdh.cSDH.GetVelocityProfile (*self*)

Get the type of velocity profile used in the SDH.

Returns

the currently set velocity profile as integer, see self.eVelocityProfileType

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Get the velocity profile from the SDH:
velocity_profile = hand.GetVelocityProfile()
# now velocity_profile is something like
# - self.eVelocityProfile["eVP_SIN_SQUARE"] == 0
# - or self.eVelocityProfile["eVP_RAMP"] == 1
```

Get the type of velocity profile used in the SDH. See html/pdf documentation for details.

10.6.3.37 def sdh.sdh.cSDH.GripHand (*self*, *grip*, *close*, *velocity*, *sequ* = `True`)

Perform one of the internal skills (a "grip").

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 a 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 fast stopped.

Parameters

self	- reference to the object itself
grip	- The index of the grip to perform [0..self.NUMBER_OF_GRIPS-1] (s.a. self.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
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.

Returns

The expected execution time for the movement in the configured time unit system uc_time.

Remarks

- Only previously enabled axes will move.
- 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 "grip" while gripping and then changes the motor current mode to "hold". After the movement previously set motor currents set for mode "move" are overwritten!

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Perform a fully opened centrical grip
hand.GripHand( self.eGraspId[ "GRIP_CENTRICAL" ], 0.0, 50.0, True )
```

Perform one of the internal skills (a "grip"). See html/pdf documentation for details.

10.6.3.38 def sdh.sdh.cSDH.IsOpen (*self*)

return wether the communication to the sdh is open or not

10.6.3.39 def sdh.sdh.cSDH.MoveAxis (*self*, *iAxis*, *sequ* = True, *check_collisions* = True)

Move one or more axes to the previously set target position with the previously set (maximum) velocities.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
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).
check_-collisions	- flag: If True (default) then collisions between the set target angles of the selected axes *iAxes* and the actual angles of the other axes are checked. If a collision is detected then the movement is **NOT** performed and a cSDHErrorInternalCollision exception is thrown. If False then no collision check is performed.

Returns

The expected 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.
- Other axes than *iAxis* will **NOT** move, even if target axis angles for their axes have been set.
- If *sequ* is True then the currently set target axis angles for other axes will be restored upon return of the function.
- If *sequ* is False then the currently set target axis angles for other axes will be overwritten with their current actual axis angles

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set a new target pose for axis 0:
hand.SetAxisTargetAngle( 0, 0 )

# Set a new target pose for axis 1,2 and 3
hand.SetAxisTargetAngle( [1, 2, 3], [-10,-20,-30] )

# Move axis 0 only
hand.MoveAxis( 0, True )
# The axis 0 has been moved to 0.0

# The target poses for axis 1,2 and 3 are still set since the
# last MoveAxis() call was sequentially.
# So move axis 1 and 2 now:
t = hand.MoveAxis( [1,2], False )
```

```
# wait until the non-sequential call has finished:
sdh.time.sleep( t )

# The axis 1 has been moved to -10 and axis 2 to -20

# The target angles for axis 3 have been overwritten since the
# last MoveAxis() call was non-sequentially.
```

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

Move an axis or some axes to the previously set target positions with the previously set (maximum) target velocities. See html/pdf documentation for details.

10.6.3.40 def sdh.sdh.cSDH.MoveFinger (*self*, *iFinger*, *sequ* = `True`, *check_collisions* = `True`)

Move a single finger to the previously set target position with the previously set (maximum) velocities.

Parameters

self	- reference to the object itself
iFinger	- index of the finger to move
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).
check_-collisions	- flag: If True (default) then collisions between the given target angles of finger *iFinger* and the actual angles of the other fingers are checked. If a collision is detected then the movement is **NOT** performed and a cSDHErrorInternalCollision exception is thrown. If False then no collision check is performed.

Returns

The expected execution time for the movement in the configured time unit system uc_time

Remarks

- The finger (i.e. all its axes) must be enabled to make the axes move
- Currently the actual movement velocity of an axis is determined by the SDH firmware to make the movements of all involved axes start and end syn-

chronously at the same time. Therefore the axis that needs the longest time for its movement at its given maximum velocity determines the velocities of all the other axes.

- 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 idependent 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

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set a new target pose for finger 0:
hand.SetFingerTargetAngle( 0, [0,0,0] )

# Set a new target pose for finger 1
hand.SetFingerTargetAngle( 1, [0,-10,-10] )

# Set a new target pose for finger 2
hand.SetFingerTargetAngle( 2, [20,-20,-20] )

# Move finger 0 only (and finger 2 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:
t = hand.MoveFinger( 1, False )

# wait until the non-sequential call has finished:
sdh.time.sleep( 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.
```

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

Move a finger to the previously set target positions with the previously set (maximum) target velocities. See html/pdf documentation for details.

10.6.3.41 def sdh.sdh.cSDH.MoveHand (*self*, *iFinger* = `All`, *sequ* = `True`, *check_collisions* = `True`)

Move all selected fingers to the previously set target position with the previously set (maximum) velocities.

Parameters

self	- reference to the object itself
iFinger	- Indices of the finger to move. Default: All fingers This can be All, a single index or a vector of indices.
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).
check_-collisions	- flag: If True (default) then collisions between the given target angles of finger *iFinger* and the actual angles of the other fingers are checked. If a collision is detected then the movement is **NOT** performed and a cSDHErrorInternalCollision exception is thrown. If False then no collision check is performed.

Returns

The expected execution time for the movement in the configured time unit system uc_time

Remarks

- Only previously enabled axes will move.
- 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.
- As axis 0 is used by finger 0 and 2 these two fingers cannot be moved completely idependent of each other. Therefor these fingers might move even if not selected.
- 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

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set a new target pose for finger 0:
hand.SetFingerTargetAngle( 0, [0,0,0] )

# Set a new target pose for finger 1
```

```
hand.SetFingerTargetAngle( 1, [0,-10,-10] )

# Set a new target pose for finger 2
hand.SetFingerTargetAngle( 2, [20,-20,-20] )

# Move fingers 0 and 2 to their target positions
hand.MoveHand( [0,2], True )
# The finger 0 has been moved to [20,0,0] and
# finger 2 to [20,-20,-20]
# (axis 0 is 'wrong' for finger 0 since the target angle for
#  axis 0 has been overwritten while setting the target angles
#  for finger 2)

# The target poses for finger 1 are still set since the
# last MoveHand() call was sequentially.
# So move finger 1 now:
t = hand.MoveHand( 1, False )

# Wait until the non-sequential call has finished:
sdh.time.sleep( t )
# The finger 1 has been moved to [0,-10,-10].


# Set new target angles for all axes ("home position")
hand.SetAxisTargetAngle( All, [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ] )

# To move all axes back to home position:
hand.MoveHand()
```

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.
=> Resolved in SDH firmware 0.0.2.7

Move selected fingers to their previously set target positions with the previously set (maximum) target velocities. See html/pdf documentation for details.

10.6.3.42 def sdh.sdh.cSDH.Open (*self*, *options* = None)

Open connection to SDH according to *options*.

Parameters

self	- reference to the object itself

options	- a collection of additional settings, like returned e.g. from cSDHOptionParser.parse_args() • Settings used by the base class cSDHBase: – `"debug_level"` : The level of debug messages to print ∗ 0: (default) no messages ∗ 1: messages of the internal cSDHSerial instance • Settings used by the internal cSDHSerial instance: – `"port"` : if set, then it is used as the port number or the device name of the serial port to use. The default value port=0 refers to 'COM1' in Windows and to the corresponding '/dev/ttyS0' in Linux. – `"timeout"` : the timeout to use: ∗ None : wait forever ∗ T : wait for T seconds (float accepted)

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Open connection to SDH according to options given to constructor:
hand.Open()

# Or use a different RS232 port 2 = COM3:
hand.Open( options=dict( port=2 ) )

# Or use settings from the command line:
parser = sdh.cSDHOptionParser( usage    =  "YOUR PROGRAM DESCRIPTION HERE" +
  "\nusage: %prog [options]",
                               revision =  "YOUR PROGRAM VERSION HERE" )
(options, args) = parser.parse_args()
hand.Open( options=options )
```

Open communication to the SDH according to options. See html/pdf documentation for details.

10.6.3.43 def sdh.sdh.cSDH.OpenRS232 (*self*, *options* = `None`)

Alias for Open() function for compatibility reasons.

Deprecated, use Open() instead.

10.6.3.44 def sdh.sdh.cSDH.SetAxisEnable (*self*, *iAxis* = `All`, *state* = `True`)

Set enabled/disabled state of axis controller(s).

The controllers of the selected axes are enabled/disabled in the SDH.

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
state	- flag: the enabled/disabled state to set This can be a single number/bool or a vector of numbers/bools.

Remarks

- Only enabled axes can move.
- Disabled axes are not powered and thus might not remain in their current position due to gravity, inertia or other external influences.
- If both *iAxis* and *state* are vectors then the order of their elements must match, i.e. `state`[i] will be applied to axis `iAxis`[i] (not axis `i`).

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Enable all axes:
hand.SetAxisEnable( sdh.All, True )

# Disable all axes:
hand.SetAxisEnable( state = 0 )

# Enable axis 0 and 2 while disabling axis 4:
hand.SetAxisEnable( [0,2,4], (True,1,False) )

# Enable axis 0:
hand.SetAxisEnable( 0 )
```

Set enabled/disabled state of axis/axes. See html/pdf documentation for details.

10.6.3.45 def sdh.sdh.cSDH.SetAxisMotorCurrent (*self*, *iAxis* = `All`, *motor_current* = `None`, *mode* = `0`)

Set the maximum allowed motor current(s) for axis(axes).

The maximum allowed motor currents are stored in the SDH. The motor currents can be stored:

- axis specific
- mode specific (see eMotorCurrentMode):
  - move : (default) The motor currents used while "moving" with a MoveHand() or MoveFinger() command (These will be overwritten by the "hold" motor currents after a GripHand() command
  - grip : The motor currents used while "gripping" with a GripHand() command
  - hold : The motor currents used after "gripping" with a GripHand() command (i.e. "holding")

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
motor_-current	- the motor current to set or `None` to keep the currently set axis motor current. This can be a single number or a vector of numbers. 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

- If both *iAxis* and *motor_current* are vectors then the order of their elements must match, i.e. `motor_current`[i] will be applied to axis `iAxis`[i] (not axis `i`)
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given).

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set maximum allowed motor current of axis 3 to 0.75 A in mode "move":
hand.SetAxisMotorCurrent( 3, 0.75, hand.eMotorCurrentMode["move"] )

# Set maximum allowed motor current of axis 3 to 0.75 A and axis 5 to 0.5 A i
  n mode "grip":
hand.SetAxisMotorCurrent( [3,5], [0.75, 0.5], hand.eMotorCurrentMode["grip"]
   )

# Set maximum allowed motor current of all axes to 1.0 A in mode "hold":
hand.SetAxisMotorCurrent( sdh.All, 1.0, hand.eMotorCurrentMode["hold"]   )

# Set maximum allowed motor current of all axes to the given values in mode "
  move"::
hand.SetAxisMotorCurrent( motor_current=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 ]
  )

# Set maximum allowed motor current of all axes in mode "move" to the current
   axis motor currents in mode "move":
hand.SetAxisMotorCurrent( motor_current=None )

# Set maximum allowed motor current of axis 3 to 0.9 A and axis 1 to its curr
  ent motor current, all in mode "hold"
hand.SetAxisMotorCurrent( [3,1], motor_current=[1.0,None], hand.eMotorCurrent
  Mode["hold"] )
```

Set the maximum allowed motor current(s) for axis(axes)

10.6.3.46 def sdh.sdh.cSDH.SetAxisTargetAcceleration (*self*, *iAxis* = `All`, *acceleration* = `None`)

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 effect 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(), SetAxisTargetGetAxisActualVelocity()

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
acceleration	- the acceleration to set or `None` to keep the currently set target acceleration of the axis This can be a single number or a vector of numbers. The value(s) are expected in the configured angular acceleration unit system uc_angular_acceleration.

Remarks

- Setting the target acceleration will **not** make the axis/axes move.
- If both *iAxis* and *acceleration* are vectors then the order of their elements must match, i.e. `acceleration`[i] will be applied to axis `iAxis`[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis acceleration of axis 3 to 100 deg/(s*s)
hand.SetAxisTargetAcceleration( 3, 100 )

# Set target axis acceleration of axis 3 to 300 deg/(s*s) and axis 5 to 350 d
  eg/(s*s)
hand.SetAxisTargetAcceleration( [3,5], [300, 350] )

# Set target axis acceleration of all axes to 111
hand.SetAxisTargetAcceleration( sdh.All, 111 )

# Set target axis acceleration of all axes to the given values
hand.SetAxisTargetAcceleration( acceleration=[100, 101, 102, 103, 104, 105, 1
  06 ] )

# Set target axis acceleration of all axes to the currently set axis accelera
  tions (well, this is not very usefull...)
hand.SetAxisTargetAcceleration( acceleration=None )

# Set target axis acceleration of axis 3 to 333 deg/(s*s) and keep axis 1 at
```

```
  its currently set target acceleration
hand.SetAxisTargetAcceleration( [3,1], acceleration=[333,None] )
```

Set the target acceleration(s) for axis(axes)

10.6.3.47 def sdh.sdh.cSDH.SetAxisTargetAngle (*self*, *iAxis* = `All`, *angle* = `None`)

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

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
angle	- the angle to set or `None` to set the current actual axis angle 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 angle will **not** make the axis/axes move.
- If both *iAxis* and *angle* are vectors then the order of their elements must match, i.e. `angle`[i] will be applied to axis `iAxis`[i] (not axis `i`)
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given).

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis angle of axis 3 to 42 deg
hand.SetAxisTargetAngle( 3, 42 )

# Set target axis angle of axis 3 to 42 deg and axis 5 to 47.11 deg
hand.SetAxisTargetAngle( [3,5], [42, 47.11] )

# Set target axis angle of all axes to 08.15
hand.SetAxisTargetAngle( sdh.All, 08.15 )

# Set target axis angle of all axes to the given values
hand.SetAxisTargetAngle( angle=[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 ] )

# Set target axis angle of all axes to the current actual axis angles
hand.SetAxisTargetAngle( angle=None )

# Set target axis angle of axis 3 to 42 deg and axis 1 to its current actual
  axis angle
hand.SetAxisTargetAngle( [3,1], angle=[42,None] )
```

Set the target angle(s) for axis(axes)

10.6.3.48 def sdh.sdh.cSDH.SetAxisTargetGetAxisActualAngle (*self*, *iAxis* = `All`, *angle* = `None`)

Set the target angle(s) and read the actual angle(s) for axis(axes).

Opposed to SetAxisTargetAngle() this will make the fingers move to the set target angles immediately, if the axis controllers are already enabled!

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
angle	- the angle to set or `None` to set the current actual axis angle 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.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes actual angles is returned
- The value(s) are returned in the configured angle unit system uc_angle.

Remarks

- Setting the target angle will **not** make the axis/axes move.
- If both *iAxis* and *angle* are vectors then the order of their elements must match, i.e. `angle`[i] will be applied to axis `iAxis`[i] (not axis `i`)
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given).

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis angle of axis 3 to 42 deg
hand.SetAxisTargetAngle( 3, 42 )

# Set target axis angle of axis 3 to 42 deg and axis 5 to 47.11 deg
hand.SetAxisTargetAngle( [3,5], [42, 47.11] )

# Set target axis angle of all axes to 08.15
hand.SetAxisTargetAngle( sdh.All, 08.15 )

# Set target axis angle of all axes to the given values
hand.SetAxisTargetAngle( angle=[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 ] )

# Set target axis angle of all axes to the current actual axis angles
hand.SetAxisTargetAngle( angle=None )

# Set target axis angle of axis 3 to 42 deg and axis 1 to its current actual
  axis angle
hand.SetAxisTargetAngle( [3,1], angle=[42,None] )
```

Set the target angle(s) and get the actual angle(s) for axis(axes)

10.6.3.49 def sdh.sdh.cSDH.SetAxisTargetGetAxisActualVelocity (*self*, *iAxis* = `All`, *velocity* = `None`)

Set the target velocity(s) and get actual velocity(s) of axis(axes).

The target velocities are stored in the SDH. The time at which a new target velocity will take effect depends on the current axis controller type:

- in eCT_POSE controller type the new target velocities will not take effect until an additional move command is sent: MoveAxis(), MoveFinger(), MoveHand()
- in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller type the new target velocity will take effect immediately, if the axis controllers are already enabled. This means that in eCT_VELOCITY_ACCELERATION controller type the accelerations must be set with SetAxisTargetAcceleration() **before** calling SetAxisTargetVelocity().

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
velocity	- the velocity to set or `None` to keep the currently set target velocity of the axis This can be a single number or a vector of numbers. The value(s) are expected in the configured angular velocity unit system uc_angular_velocity.

Returns

- if *iAxis* is a single index then a single float value is returned
- else a list of the selected axes actual velocities is returned
- The value(s) are reported in the configured angular velocity unit system uc_angular_velocity.

Remarks

- If both *iAxis* and *velocity* are vectors then the order of their elements must match, i.e. `velocity`[i] will be applied to axis `iAxis`[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis velocity of axis 3 to 14 deg/s
hand.SetAxisTargetGetAxisActualVelocity( 3, 14 )

# Set target axis velocity of axis 3 to 14 deg/s and axis 5 to 12.34 degs
hand.SetAxisTargetGetAxisActualVelocity( [3,5], [14, 12.34] )

# Set target axis velocity of all axes to 08.15 deg/s
hand.SetAxisTargetGetAxisActualVelocity( sdh.All, 08.15 )
```

```
# Set target axis velocity of all axes to the given values
hand.SetAxisTargetGetAxisActualVelocity( velocity=[10.0, 20.0, 30.0, 40.0, 50
  .0, 60.0, 70.0 ] )

# Set target axis velocity of all axes to the currently set axis velocities (
  well, this is not very usefull...)
hand.SetAxisTargetGetAxisActualVelocity( velocity=None )

# Set target axis velocity of axis 3 to 14 deg/s and keep axis 1 at its curre
  ntly set target velocity
hand.SetAxisTargetGetAxisActualVelocity( [3,1], velocity=[14,None] )
```

Set the target velocity(s) and get the actual velocity(s) for axis(axes)

10.6.3.50 def sdh.sdh.cSDH.SetAxisTargetVelocity (*self*, *iAxis* = `All`, *velocity* = `None`)

Set the target velocity(s) for axis(axes).

The target velocities are stored in the SDH. The time at which a new target velocity will take effect depends on the current axis controller type:

- in eCT_POSE controller type the new target velocities will not take effect until an additional move command is sent: MoveAxis(), MoveFinger(), MoveHand()
- in eCT_VELOCITY and eCT_VELOCITY_ACCELERATION controller type the new target velocity will take effect immediately, if the axis controllers are already enabled. This means that in eCT_VELOCITY_ACCELERATION controller type the accelerations must be set with SetAxisTargetAcceleration() **before** calling SetAxisTargetVelocity().

Parameters

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
velocity	- the velocity to set or `None` to keep the currently set target velocity of the axis This can be a single number or a vector of numbers. The value(s) are expected in the configured angular velocity unit system uc_angular_-velocity.

Remarks

- If both *iAxis* and *velocity* are vectors then the order of their elements must match, i.e. `velocity`[i] will be applied to axis `iAxis`[i] (not axis `i`).
- If *iAxis* is `All` then the order will be the natural one (as if *iAxis*=[0,1,2,3,4,5,6] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis velocity of axis 3 to 14 deg/s
```

```
hand.SetAxisTargetVelocity( 3, 14 )

# Set target axis velocity of axis 3 to 14 deg/s and axis 5 to 12.34 degs
hand.SetAxisTargetVelocity( [3,5], [14, 12.34] )

# Set target axis velocity of all axes to 08.15 deg/s
hand.SetAxisTargetVelocity( sdh.All, 08.15 )

# Set target axis velocity of all axes to the given values
hand.SetAxisTargetVelocity( velocity=[10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.
  0 ] )

# Set target axis velocity of all axes to the currently set axis velocities (
  well, this is not very usefull...)
hand.SetAxisTargetVelocity( velocity=None )

# Set target axis velocity of axis 3 to 14 deg/s and keep axis 1 at its curre
  ntly set target velocity
hand.SetAxisTargetVelocity( [3,1], velocity=[14,None] )
```

Set the target velocity(s) for axis(axes)

10.6.3.51 def sdh.sdh.cSDH.SetController (*self*, *controller*)

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 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.

Parameters

self	- reference to the object itself
controller	- identifier of controller to set. Valid values are defined in self.eControllerType

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

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set the default coordinated position controller in the SDH:
# (see e.g. demo-simple.cpp, demo-simple2.cpp, demo-simple3.cpp for further e
  xamples)
hand.SetController( hand.eControllerType.eCT_POSE )

# Set the velocity controller in the SDH:
hand.SetController( hand.eControllerType.eCT_VELOCITY )

# Or set the velocity controller using a string parameter:
# (see e.g. demo-velocity-acceleration.cpp for further examples)
hand.SetController( "eCT_VELOCITY_ACCELERATION" )
```

Set the type of axis controller to be used in the SDH. See html/pdf documentation for details.

10.6.3.52 def sdh.sdh.cSDH.SetFingerEnable (*self*, *iFinger* = `All`, *state* = `True`)

Set enabled/disabled state of axis controllers of finger(s).

Parameters

self	- reference to the object itself
iFinger	- index of finger to access. This can be All, a single index or a vector of indices.
state	- flag: the enabled/disabled state to set This can be a single number/bool or a vector of numbers/bools.

Remarks

- Only enabled fingers can move.
- The axes of disabled fingers are not powered and thus might not remain in their current position due to gravity, inertia or other external influences.
- As axis 0 is used for finger 0 and 2, axis 0 is disabled only if both finger 0 and 1 are disabled.
- If both *iFinger* and *state* are vectors then the order of their elements must match, i.e. `state`[i] will be applied to all axes of finger `iFinger`[i] (not finger `i`).
- If *iFinger* is `All` then the order will be the natural one (as if *iFinger*=[0,1,2] had been given

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Enable all fingers:
hand.SetFingerEnable( sdh.All, True )

# Disable all fingers:
hand.SetFingerEnable( state = 0 )

# Enable finger 1 and 2 while disabling finger 0 :
hand.SetFingerEnable( sdh.All, (False,True, True) )
# (this will keep axis 0 (used by finger 0) enabled, as axis 0 is needed by f
  inger 2 too)

# Disable finger 2:
hand.SetFingerEnable( 2, False )

# Enable fingers 0 and 2
hand.SetFingerEnable( [0,2], [True,1] )
```

Set enabled/disabled state of finger(s). See html/pdf documentation for details.

10.6.3.53 def sdh.sdh.cSDH.SetFingerTargetAngle (*self*, *iFinger*, *angle* = `None`)

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

self	- reference to the object itself
iFinger	- index of finger to access. This must be a single index.
angle	- 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.

Examples:

```
# Assuming "hand" is a sdh.cSDH object ...

# Set target axis angles of finger 0 to [ 10.0, -08.15, 47.11 ]
hand.SetFingerTargetAngle( 0, [ 10.0, -08.15, 47.11 ] )

# Set target axis angles of finger 1 to [ 0.0, 24.7, 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 )

# Setting target axis angles of all axes of finger 1 to 42.0
# would result in cSDHErrorInvalidParameter exception since the virtual
# axis 0 of finger 1 can only be set to 0.0
```

```
# Set target axis angles of all axes of finger 2 to their current actual angl
  es
hand.SetFingerTargetAngle( 2, None )
```

Set the target axis angles for a single finger. See html/pdf documentation for details.

10.6.3.54 def sdh.sdh.cSDH.SetVelocityProfile (*self, velocity_profile*)

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

Parameters

self	- reference to the object itself
velocity_-profile	- Name or number of velocity profile to set. Valid values are defined in self.eVelocityProfileType

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...

# Set the sin square velocity profile in the SDH:
hand.SetVelocityProfile( hand.eVelocityProfile.eVP_SIN_SQUARE )

# Or else set the ramp velocity profile in the SDH:
hand.SetVelocityProfile( hand.eVelocityProfile.eVP_RAMP )

# Or else set the ramp velocity profile using a string:
hand.SetVelocityProfile( "eVP_RAMP" )
```

Set the type of velocity profile to be used in the SDH. See html/pdf documentation for details.

10.6.3.55 def sdh.sdh.cSDH.Stop (*self*)

Stop movement of all axes but keep controllers on.

This command will always be executed sequentially: it will return only after the SDH has performed and 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.
=> Resolved in SDH firmware 0.0.2.7

Examples:

```
# Assuming 'hand' is a sdh.cSDH object ...
```

```
# Perform a stop:
hand.Stop()
```

Stop movement of all axes of the SDH. See html/pdf documentation for details.

10.6.3.56 def sdh.sdh.cSDH.UseDegrees (*self*)

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 sdh.cSDH object ...

# make hand object use degrees and degrees/second for angles and angular velo
  cities
hand.UseDegrees()
# as degrees, degrees/second are the default this is needed only if the
# unit system was changed before
```

Shortcut to set the unit system to degrees.

10.6.3.57 def sdh.sdh.cSDH.UseRadians (*self*)

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 sdh.cSDH object ...

# make hand object use radians and radians/second for angles and angular velo
  cities
hand.UseRadians()
```

Shortcut to set the unit system to radians.

10.6.3.58 def sdh.sdh.cSDH.WaitAxis (*self*, *iAxis* = `All`, *timeout* = `None`)

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

self	- reference to the object itself
iAxis	- index of axis to access. This can be All, a single index or a vector of indices.
timeout	- a timeout in seconds or None (default).

Remarks

- If timeout is None is given then this function will wait arbitrarily long
- If a timeout is given then this function will raise a cSDHErrorTimeout exception if the given axes are still moving after timeout many seconds

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 sdh.cSDH object ...

hand.SetController( eCT_POSE );

# Set a new target pose for axis 1,2 and 3
hand.SetAxisTargetAngle( [1, 2, 3], [-10,-20,-30] )

# Move axes there non sequentially:
hand.MoveAxis( [1, 2, 3], 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( [1, 2, 3] )


# go back home (all angles to 0.0):
hand.SetAxisTargetAngle( sdh.All, 0.0 )

# Move all axes there non sequentially:
hand.MoveAxis( sdh.All, False )

# Wait until all axes are there:
hand.WaitAxis()
# 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 sdh.cSDH object ...

hand.SetController( eCT_VELOCITY_ACCELERATION );

# Set a new target pose for axis 1,2 and 3
hand.SetAxisTargetVelocity( [1, 2, 3], [-10,-20,-30] ) # will make the axes m
  ove!

# 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
hand.SetAxisTargetVelocity( [1, 2, 3], [0,0,0] ) # will make the axes stop wi
  th the default (de)acceleration

# The previous command returned immediately, so
# before doing something else with the hand make shure the
# selected axes have stopped:
hand.WaitAxis( [1,2,3] );

# now the axes have stopped
```

Wait until the axis(axes) have actually finished their movement

10.6.4 Member Data Documentation

10.6.4.1 sdh.sdh.cSDH.controller_type

10.6.4.2 sdh.sdh.cSDH.eMotorCurrentMode

the motor current can be set specifically for these modes:

10.6.4.3 sdh.sdh.cSDH.f_eps_a

array of 3 epsilon values

10.6.4.4 sdh.sdh.cSDH.f_max_acceleration_a

Maximum allowed axis acceleration (in internal units (degrees/second²)), including the virtual axis.

10.6.4.5 sdh.sdh.cSDH.f_max_angle_a

Maximum allowed axis angles (in internal units (degrees)), including the virtual axis.

10.6.4.6 sdh.sdh.cSDH.f_max_motor_current_a

Maximum allowed motor currents (in internal units (Ampere)), including the virtual axis.

10.6.4.7 sdh.sdh.cSDH.f_max_velocity_a

Maximum allowed axis velocity (in internal units (degrees/second)), including the virtual axis we cannot read the real limits from the SDH firmware yet, since we are not yet connected.

10.6.4.8 sdh.sdh.cSDH.f_min_acceleration_a

Minimum allowed axis acceleration (in internal units (degrees/second²)), including the virtual axis.

10.6.4.9 sdh.sdh.cSDH.f_min_angle_a

Minimum allowed axis angles (in internal units (degrees)), including the virtual axis.

10.6.4.10 sdh.sdh.cSDH.f_min_velocity_a

Minimum allowed axis velocity (in internal units (degrees/second)), including the virtual axis we cannot read the real limits from the SDH firmware yet, since we are not yet connected.

10.6.4.11 sdh.sdh.cSDH.f_ones_a

array of 3 1 values

10.6.4.12 sdh.sdh.cSDH.f_zeros_a

array of 3 0 values

10.6.4.13 sdh.sdh.cSDH.finger_axis_index

Mapping of finger index, finger axis index to axis index:

10.6.4.14 sdh.sdh.cSDH.finger_number_of_axes

Mapping of finger index to number of real axes of fingers:

10.6.4.15 sdh.sdh.cSDH.grip_max_velocity

Maximum allowed grip velocity (in internal units (degrees/second))

10.6.4.16 sdh.sdh.cSDH.interface

The interface to the SDH hardware:

10.6.4.17 sdh.sdh.cSDH.l1

length of limb 1 (proximal joint to distal joint) in mm

10.6.4.18 sdh.sdh.cSDH.l2

length of limb 2 (distal joint to fingertip) in mm

10.6.4.19 sdh.sdh.cSDH.max_angular_acceleration_a

Maximum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

Reimplemented from sdh.sdhbase.cSDHBase.

10.6.4.20 sdh.sdh.cSDH.max_angular_velocity_a

Maximum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

Reimplemented from sdh.sdhbase.cSDHBase.

10.6.4.21 sdh.sdh.cSDH.min_angular_velocity_a

Minimum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

Reimplemented from sdh.sdhbase.cSDHBase.

10.6.4.22 sdh.sdh.cSDH.NUMBER_OF_AXES_PER_FINGER

The number of axis per finger (for finger 1 this includes the "virtual" base axis)

10.6.4.23 sdh.sdh.cSDH.NUMBER_OF_VIRTUAL_AXES

The number of virtual axes.

10.6.4.24 sdh.sdh.cSDH.offset

list of xyz-arrays for all fingers with offset from (0,0,0) of proximal joint in mm x, y, z

10.6.4.25 sdh.sdh.cSDH.release_firmware

10.6.4.26 sdh.sdh.cSDH.uc_angle

unit convert for (axis) angles: default = unit.uc_angle_degrees

10.6.4.27 sdh.sdh.cSDH.uc_angular_acceleration

unit convert for (axis) angular accelerations: default = unit.uc_angular_acceleration_-degrees_per_second_squared

10.6.4.28 sdh.sdh.cSDH.uc_angular_velocity

unit convert for (axis) angular velocities: default = unit.uc_angular_velocity_degrees_-per_second

10.6.4.29 sdh.sdh.cSDH.uc_motor_current

unit converter for motor curent: default = unit.uc_motor_current_ampere

10.6.4.30 sdh.sdh.cSDH.uc_position

unit converter for position: default = unit.uc_position_millimeter

10.6.4.31 sdh.sdh.cSDH.uc_temperature

unit convert for temperatures: default = unit.uc_temperature_celsius

10.6.4.32 sdh.sdh.cSDH.uc_time

unit convert for times: default = unit.uc_time_seconds

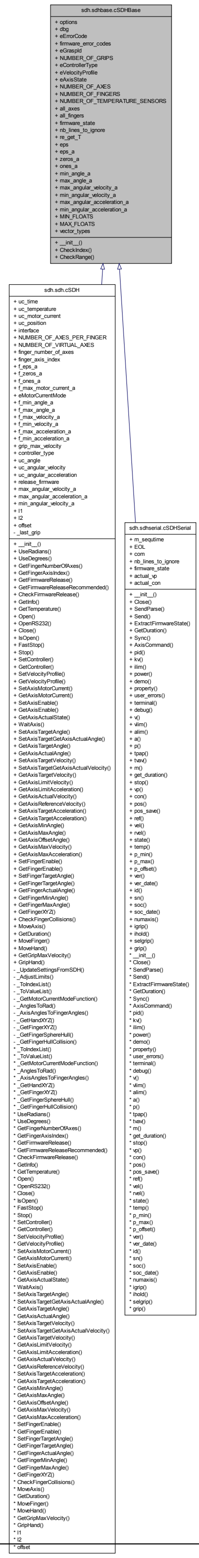
The documentation for this class was generated from the following file:

- sdh/sdh.py

10.7 sdh.sdhbase.cSDHBase Class Reference

The base class to control the SCHUNK Dexterous Hand.

Inheritance diagram for sdh.sdhbase.cSDHBase:



Public Member Functions

- def __init__

  Constructor of cSDHBase class, initilize internal variables and settings.

- def CheckIndex

  Check if index is in [0 .

- def CheckRange

  Check if value is in [minvalue .

Public Attributes

- options
- dbg

  tDBG object to disable/enable debug messages on demand

- eErrorCode

  A dictionary to store the enum values of the error codes of the SDH firmware.

- firmware_error_codes

  A dictinary to map error codes to human readable error messages.

- eGraspId

  A lightweight object to store the enum values of the known grasps.

- NUMBER_OF_GRIPS

  The number of valid grips.

- eControllerType

  A dictionary to store the enum values of the controller types.

- eVelocityProfile

  A dictionary to store the enum values of the velocity profile types of the SDH firmware.

- eAxisState

  A dictionary to store the enum values of the axis states of the SDH firmware.

- NUMBER_OF_AXES

  The number of axes.

- NUMBER_OF_FINGERS

  The number of fingers.

- NUMBER_OF_TEMPERATURE_SENSORS

The number of temperature sensors.

- all_axes

  A list with all axis indices [0,1,...,NUMBER_OF_AXES-1].

- all_fingers

  A list with all finger indices [0,1,...,NUMBER_OF_FINGERS-1].

- firmware_state

  the last known state of the SDH firmware

- nb_lines_to_ignore

  number of remaining reply lines of a previous (non sequential) command

- re_get_T

  regular expression to extract the duration of a command from its reply

- eps

  epsilon value (max absolute deviation of reported values from actual hardware values) (needed since firmware limits number of digits reported)

- eps_a

  array of 7 epsilon values

- zeros_a

  array of 7 0 values

- ones_a

  array of 7 1 values

- min_angle_a

  Minimum allowed axis angles (in internal units (degrees))

- max_angle_a

  Maximum allowed axis angles (in internal units (degrees))

- max_angular_velocity_a

  Maximum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

- min_angular_velocity_a

  Minimum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

- max_angular_acceleration_a

Maximum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

- min_angular_acceleration_a

  Minimum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

- MIN_FLOATS

  array of 7 MIN_FLOAT values

- MAX_FLOATS

  array of 7 MAX_FLOAT values

- vector_types

  A list with all vector-like types that are accepted as parameters.

10.7.1 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.

The base class to control the SCHUNK Dexterous Hand. See html/pdf documentation for details.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 def sdh.sdhbase.cSDHBase.__init__ (*self*, *options* = `None`)

Constructor of cSDHBase class, initilize internal variables and settings.

Parameters

self	- reference to the object itself
options	- a dictionary of additional settings, like the options.__dict__ returned from cSDHOptionParser.parse_args() • `"debug_level"` : if set, then it is used as debug level of the created object, else a default of 0 is used. 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.)

Constructor of cSDHBase class, initilize internal variables and settings

Reimplemented in sdh.sdh.cSDH, and sdh.sdhserial.cSDHSerial.

10.7.3 Member Function Documentation

10.7.3.1 def sdh.sdhbase.cSDHBase.CheckIndex (*self*, *index*, *maxindex*, *name* = " ")

Check if *index* is in [0 .

. *maxindex-1*] or All. Raise a cSDHErrorInvalidParameter exception if not.

10.7.3.2 def sdh.sdhbase.cSDHBase.CheckRange (*self*, *value*, *minvalue*, *maxvalue*, *name* = " ")

Check if *value* is in [*minvalue* .

. *maxvalue*]. Raise a cSDHErrorInvalidParameter exception if not.

10.7.4 Member Data Documentation

10.7.4.1 sdh.sdhbase.cSDHBase.all_axes

A list with all axis indices [0,1,...,NUMBER_OF_AXES-1].

10.7.4.2 sdh.sdhbase.cSDHBase.all_fingers

A list with all finger indices [0,1,...,NUMBER_OF_FINGERS-1].

10.7.4.3 sdh.sdhbase.cSDHBase.dbg

tDBG object to disable/enable debug messages on demand

10.7.4.4 sdh.sdhbase.cSDHBase.eAxisState

A dictionary to store the enum values of the axis states of the SDH firmware.

10.7.4.5 sdh.sdhbase.cSDHBase.eControllerType

A dictionary to store the enum values of the controller types.

10.7.4.6 sdh.sdhbase.cSDHBase.eErrorCode

A dictionary to store the enum values of the error codes of the SDH firmware.

10.7.4.7 sdh.sdhbase.cSDHBase.eGraspId

A lightweight object to store the enum values of the known grasps.

10.7.4.8 sdh.sdhbase.cSDHBase.eps

epsilon value (max absolute deviation of reported values from actual hardware values) (needed since firmware limits number of digits reported)

10.7.4.9 sdh.sdhbase.cSDHBase.eps_a

array of 7 epsilon values

10.7.4.10 sdh.sdhbase.cSDHBase.eVelocityProfile

A dictionary to store the enum values of the velocity profile types of the SDH firmware.

10.7.4.11 sdh.sdhbase.cSDHBase.firmware_error_codes

A dictinary to map error codes to human readable error messages.

10.7.4.12 sdh.sdhbase.cSDHBase.firmware_state

the last known state of the SDH firmware

Reimplemented in sdh.sdhserial.cSDHSerial.

10.7.4.13 sdh.sdhbase.cSDHBase.max_angle_a

Maximum allowed axis angles (in internal units (degrees))

10.7.4.14 sdh.sdhbase.cSDHBase.max_angular_acceleration_a

Maximum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

Reimplemented in sdh.sdh.cSDH.

10.7.4.15 sdh.sdhbase.cSDHBase.max_angular_velocity_a

Maximum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

Reimplemented in sdh.sdh.cSDH.

10.7.4.16 sdh.sdhbase.cSDHBase.MAX_FLOATS

array of 7 MAX_FLOAT values

10.7.4.17 sdh.sdhbase.cSDHBase.min_angle_a

Minimum allowed axis angles (in internal units (degrees))

10.7.4.18 sdh.sdhbase.cSDHBase.min_angular_acceleration_a

Minimum allowed axis angular accelerations (in internal units (degrees/(s∗s)))

10.7.4.19 sdh.sdhbase.cSDHBase.min_angular_velocity_a

Minimum allowed axis angular velocities (in internal units (degrees/s)) these are overwritten later when connected to the real hand by reading the actual limits from the SDH firmware.

Reimplemented in sdh.sdh.cSDH.

10.7.4.20 sdh.sdhbase.cSDHBase.MIN_FLOATS

array of 7 MIN_FLOAT values

10.7.4.21 sdh.sdhbase.cSDHBase.nb_lines_to_ignore

number of remaining reply lines of a previous (non sequential) command

Reimplemented in sdh.sdhserial.cSDHSerial.

10.7.4.22 sdh.sdhbase.cSDHBase.NUMBER_OF_AXES

The number of axes.

10.7.4.23 sdh.sdhbase.cSDHBase.NUMBER_OF_FINGERS

The number of fingers.

10.7.4.24 sdh.sdhbase.cSDHBase.NUMBER_OF_GRIPS

The number of valid grips.

10.7.4.25 sdh.sdhbase.cSDHBase.NUMBER_OF_TEMPERATURE_SENSORS

The number of temperature sensors.

10.7.4.26 sdh.sdhbase.cSDHBase.ones_a

array of 7 1 values

10.7.4.27 sdh.sdhbase.cSDHBase.options

10.7.4.28 sdh.sdhbase.cSDHBase.re_get_T

regular expression to extract the duration of a command from its reply

10.7.4.29 sdh.sdhbase.cSDHBase.vector_types

A list with all vector-like types that are accepted as parameters.

Most methods of cSDHBase and derivatives accept not only single numbers, but vectors of several numbers as parameters, These types are herein refered to as **"vector"**.

TODO: This should be made more general, e.g. to work with derived classes. What we acutally need to know if the parameter is iterable, see e.g. `http://bytes.com/topic/python/answers/514838-`

10.7.4.30 sdh.sdhbase.cSDHBase.zeros_a

array of 7 0 values

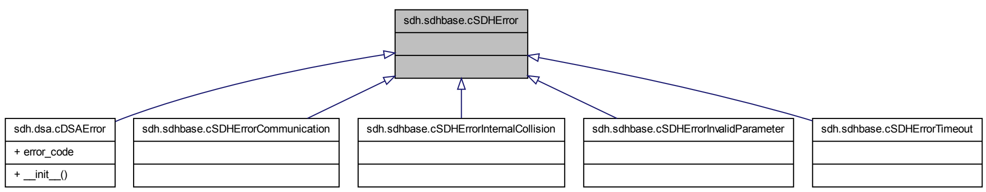
The documentation for this class was generated from the following file:

- sdh/sdhbase.py

10.8 sdh.sdhbase.cSDHError Class Reference

Exception classes.

Inheritance diagram for sdh.sdhbase.cSDHError:



10.8.1 Detailed Description

Exception classes. Base class for exceptions in the sd module

Base class for exceptions in the sd module.

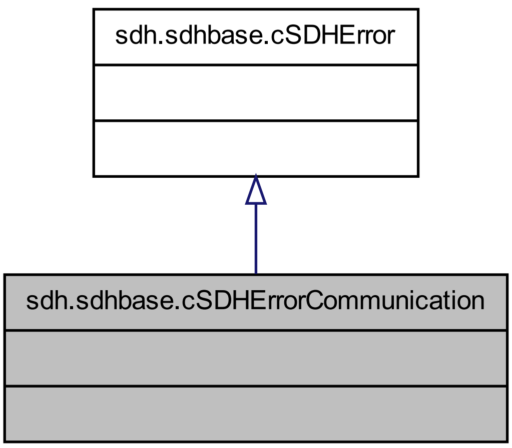
The documentation for this class was generated from the following file:

- sdh/sdhbase.py

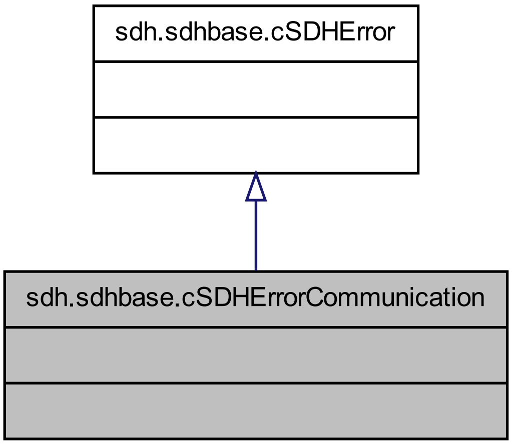
10.9 sdh.sdhbase.cSDHErrorCommunication Class Reference

SDH-exception: Communication error occured in the sd module.

Inheritance diagram for sdh.sdhbase.cSDHErrorCommunication:



Collaboration diagram for sdh.sdhbase.cSDHErrorCommunication:



10.9.1 Detailed Description

SDH-exception: Communication error occured in the sd module. Communication error occured in the sd module.

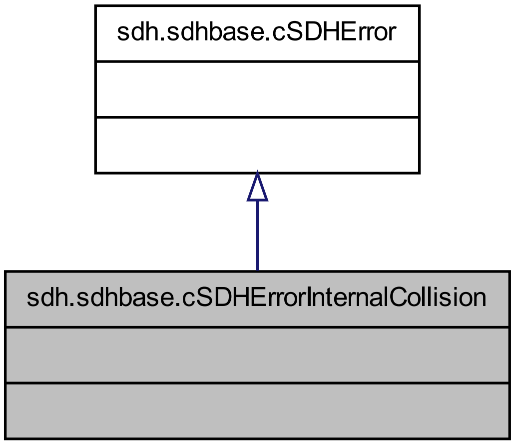
The documentation for this class was generated from the following file:

- sdh/sdhbase.py

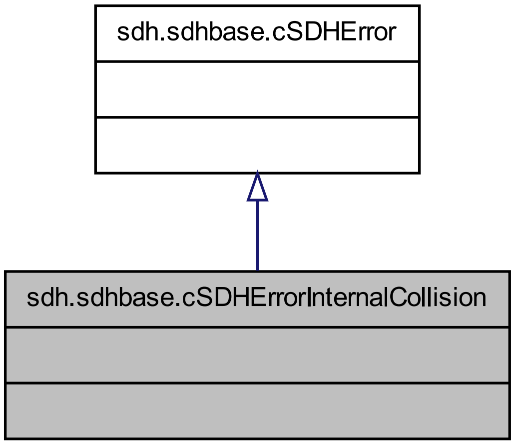
10.10 sdh.sdhbase.cSDHErrorInternalCollision Class Reference

SDH-exception: The given target angles would lead to an internal collision.

Inheritance diagram for sdh.sdhbase.cSDHErrorInternalCollision:



Collaboration diagram for sdh.sdhbase.cSDHErrorInternalCollision:



10.10.1 Detailed Description

SDH-exception: The given target angles would lead to an internal collision. The given target angles would lead to an internal collision.

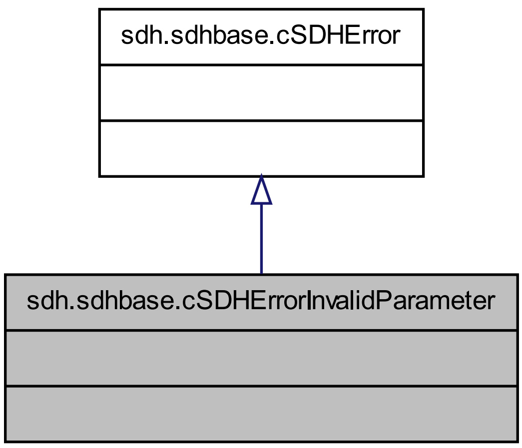
The documentation for this class was generated from the following file:

- sdh/sdhbase.py

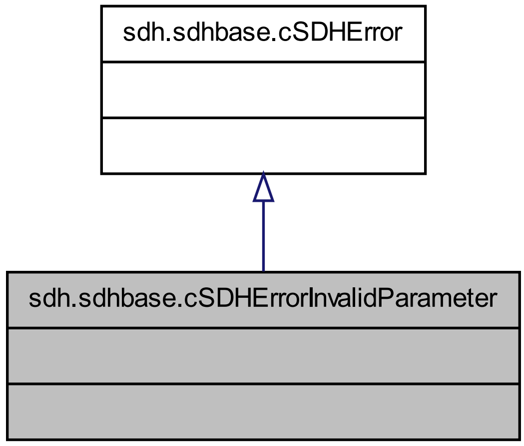
10.11 sdh.sdhbase.cSDHErrorInvalidParameter Class Reference

SDH-exception: Invalid parameter(s) were given.

Inheritance diagram for sdh.sdhbase.cSDHErrorInvalidParameter:



Collaboration diagram for sdh.sdhbase.cSDHErrorInvalidParameter:



10.11.1 Detailed Description

SDH-exception: Invalid parameter(s) were given. Invalid parameter(s) were given.

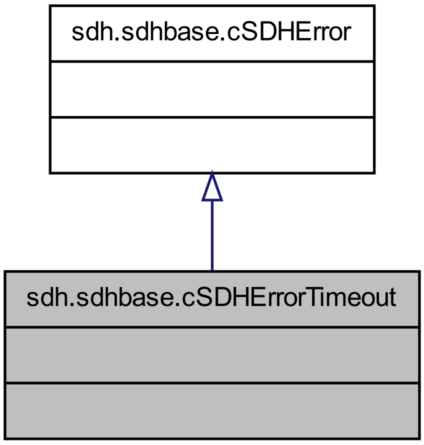
The documentation for this class was generated from the following file:

- sdh/sdhbase.py

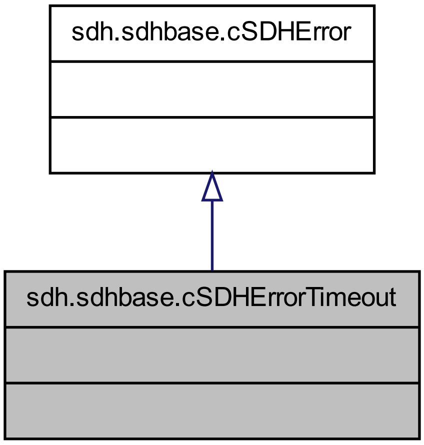
10.12 sdh.sdhbase.cSDHErrorTimeout Class Reference

SDH-exception: A (communication) timeout occured.

Inheritance diagram for sdh.sdhbase.cSDHErrorTimeout:



Collaboration diagram for sdh.sdhbase.cSDHErrorTimeout:



10.12.1 Detailed Description

SDH-exception: A (communication) timeout occured. A (communication) timeout occured.

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

- sdh/sdhbase.py

10.13 sdh.auxiliary.cSDHOptionParser Class Reference

Customized OptionParser with some SDH specific options already set.

Inherits OptionParser.

Public Member Functions

- def CBDebugLog

  Callback for the -l | --debuglog option.

- def CBTCP

  Callback for the --tcp option.

- def CBDSATCP

  Callback for the --dsa_tcp option.

- def parse_args

  overloaded version of the optparse.OptionParser.parse_args() method parse the args and perform some actions if requested (like reading version info from SDH)

- def __init__

  Create a cSDHOptionParser instance.

Public Attributes

- revision

Static Public Attributes

- int default_sdh_port = 0
- int default_dsa_port = 4
- string default_tcp_adr = "192.168.1.42"
- int default_tcp_port = 23
- int default_dsa_tcp_port = 13000

10.13.1 Detailed Description

Customized OptionParser with some SDH specific options already set. The following common options are already set

- -p | --port PORT : use com port PORT instead of the default 0 (sets options.port)
- -d | --debug LEVEL : turn on debug (sets options.debug_level to LEVEL)
- -R | --radians : Use radians and radians per second for angles and angular velocities instead of default degrees and degrees per second (sets options.use_radians)
- -F | --fahrenheit : Use degrees fahrenheit to report temperatures instead of default degrees celsius" (sets options.use_fahrenheit)
- -v | --version : print version and exit
- -T | --timeout : Timeout used to wait for answers from SDH (The default is None which means wait for ever)

An object of this class can be used by scripts that access the SDH. Such scripts can of course add further script specific options. For example uses see the demo-*.py scripts

example

```
# Command line option handling

# Create an option parser object to parse common command line options
parser = sdh.cSDHOptionParser( usage    =  __doc__ + "\nusage: %prog [options
  ]",
                               revision = __version__ )

Parse (and handle, if possible) the command line options of the script
(options, args) = parser.parse_args()

# The parsed command line arguments are now stored in the options
# object. E.g. options.port is the communication port to use, either
# the default one or the one read from the -p | --port command line
# argument:
print "The serial port %d will be used" % (options.port)

# For even more comfort the parsed options can be forwarded directly
# to a cSDH constructor, like in:
hand = sdh.cSDH( options=options.__dict__ )
```

Customized OptionParser with some SDH specific options already set. See html/pdf documentation for details.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 def sdh.auxiliary.cSDHOptionParser.__init__ (*self*, *usage* = " ", *revision* = " ")

Create a cSDHOptionParser instance.

Parameters

self	- reference to the object itself
usage	- A string describing the purpose of the calling script
revision	- A string describing the revision of the calling script

Create a cSDHOptionParser instance. See html/pdf documentation for details.

10.13.3 Member Function Documentation

10.13.3.1 def sdh.auxiliary.cSDHOptionParser.CBDebugLog (*self*, *option*, *opt_str*, *value*, *parser*, *args*, *kwarg*)

Callback for the -l | --debuglog option.

10.13.3.2 def sdh.auxiliary.cSDHOptionParser.CBDSATCP (*self*, *option*, *opt_str*, *value*, *parser*, *args*, *kwarg*)

Callback for the --dsa_tcp option.

10.13.3.3 def sdh.auxiliary.cSDHOptionParser.CBTCP (*self*, *option*, *opt_str*, *value*, *parser*, *args*, *kwarg*)

Callback for the --tcp option.

10.13.3.4 def sdh.auxiliary.cSDHOptionParser.parse_args (*self*, *args* = `None`, *values* = `None`)

overloaded version of the optparse.OptionParser.parse_args() method parse the args and perform some actions if requested (like reading version info from SDH)

10.13.4 Member Data Documentation

10.13.4.1 int sdh.auxiliary.cSDHOptionParser.default_dsa_port = 4 [static]

10.13.4.2 int sdh.auxiliary.cSDHOptionParser.default_dsa_tcp_port = 13000 [static]

10.13.4.3 int sdh.auxiliary.cSDHOptionParser.default_sdh_port = 0 [static]

10.13.4.4 string sdh.auxiliary.cSDHOptionParser.default_tcp_adr = "192.168.1.42" [static]

10.13.4.5 int sdh.auxiliary.cSDHOptionParser.default_tcp_port = 23 [static]

10.13.4.6 sdh.auxiliary.cSDHOptionParser.revision

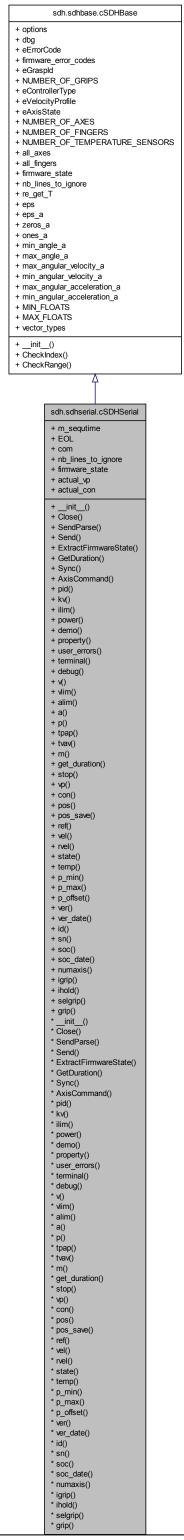
The documentation for this class was generated from the following file:

- sdh/auxiliary.py

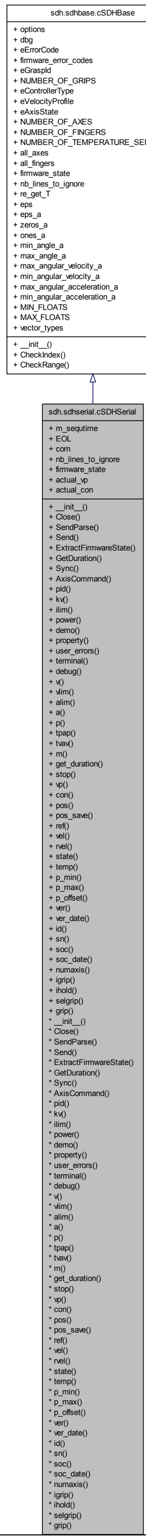
10.14 sdh.sdhserial.cSDHSerial Class Reference

The class to communicate with a SDH via RS232.

Inheritance diagram for sdh.sdhserial.cSDHSerial:



Collaboration diagram for sdh.sdhserial.cSDHSerial:



Public Member Functions

Internal methods

- def __init__
  Constructor of cSDHSerial.
- def Close
  Close connection to serial interface.
- def SendParse
  Simplified parsing of 1 line commands.
- def Send
  Send command string s+EOL to self.com and read reply according to nb_lines.
- def ExtractFirmwareState
  Try to extract the state of the SDH firmware from the lines reply.
- def GetDuration
  Return duration of the execution of a SDH command as reported by line.
- def Sync
  Read all pending lines from SDH to resync execution of PC and SDH.
- def AxisCommand
  Get/Set values.

Setup and configuration methods

- def pid
  Get/Set PID controller parameters.
- def kv
  Get/Set kv parameter.
- def ilim
  Get/Set current limit for m command.
- def power
  Get/Set current power state.

Misc. methods

- def demo
  Enable/disable SCHUNK demo.
- def property
  Set named property.

- def user_errors
- def terminal
- def debug

Movement methods

- def v
  Get/Set target velocity.
- def vlim
  Get velocity limits.
- def alim
  Get acceleration limits.
- def a
  Get/Set target acceleration.
- def p
  Get/Set target angle for axis.
- def tpap
  Set target angle, get actual angle for axis.
- def tvav
  Set target velocity, get actual velocity for axis.
- def m
  Send move command.
- def get_duration
  Send get_duration command.
- def stop
  Stop sdh.
- def vp
  Get/set velocity profile.
- def con
  Get/set controller type.

Diagnostic and identification methods

- def pos
  Get actual angle/s of axis/axes.
- def pos_save
  Save actual angle/s to non volatile memory.

- def ref
  Do reference movements with selected axes.
- def vel
  Get actual angular velocity/ies of axis/axes.
- def rvel
  Get reference angular velocity/ies of axis/axes.
- def state
  Get actual state/s of axis/axes.
- def temp
  Get actual temperatures of SDH.
- def p_min
  Get/Set minimum allowed target angle for axis.
- def p_max
  Get/Set maximum allowed target angle for axis.
- def p_offset
  Get/Set offset for axis.
- def ver
  Return version of SDH firmware.
- def ver_date
  Return date of SDH firmware.
- def id
  Return id of SDH.
- def sn
  Return sn of SDH.
- def soc
  Return soc of SDH.
- def soc_date
  Return soc of SDH.
- def numaxis
  Return number of axis of SDH.

Grip methods

- def igrip
  Get/Set motor current limits for grip commands.

- def ihold

  Get/Set motor current limits for hold commands.

- def selgrip

  Send "selgrip grip" command to SDH.

- def grip

  send "grip=close,velocity" command to SDH close : [0.0 .

Public Attributes

- m_sequtime

  additional time in seconds to wait for sequential execution of "m"-command (as these are always executed non-sequentially by the SDH firmware) (no longer needed since WaitAxis() is used to ensure movement has ended)

- EOL

  String to use as "End Of Line" marker when sending to SDH.

- com

  the RS232 connection to use for communication

- nb_lines_to_ignore

  number of remaining reply lines of a previous (non sequential) command

- firmware_state

  the last known state of the SDH firmware

- actual_vp
- actual_con

10.14.1 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.

The class to communicate with a SDH via RS232. See html/pdf documentation for details.

Bug

SCHUNK-internal bugzilla ID: Bug 1517
With SDH firmware 0.0.3.x the first connection to a newly powered up SDH can yield an error especially when connecting via TCP.
=> Resolved in SDHLibrary-python 0.0.2.8

10.14.2 Constructor & Destructor Documentation

10.14.2.1 def sdh.sdhserial.cSDHSerial.__init__ (*self*, *options* = `None`)

Constructor of cSDHSerial.

- Open the serial port
- Check connection to SDH by querying the SDH firmware version

This may raise an exception on failure

Parameters

self	- reference to the object itself

options	- a dictionary of additional settings, like the options.__dict__ returned from cSDHOptionParser.parse_args()
	• Settings used by the base class cSDHBase:
	  – `"debug_level"` : if set, then it is used as debug level of the created object, else a default of 0 is used
	• Settings used by this cSDHSerial class:
	  – `"port"`: if set, then it is used as the port number or the device name of the serial port to use. The default value port=0 refers to 'COM1' in Windows and to the corresponding '/dev/ttyS0' in Linux.
	  – `"timeout"` : the timeout to use:
	    ∗ None : wait forever
	    ∗ T : wait for T seconds (float accepted)
	• (Superclasses of cSDHSerial use additional settings, see there.)
	• (Using classes of cSDHSerial like cSDH use additional settings, see there.)

Constructor of cSDHSerial. See html/pdf documentation for details.

Reimplemented from sdh.sdhbase.cSDHBase.

10.14.3 Member Function Documentation

10.14.3.1 def sdh.sdhserial.cSDHSerial.a (*self*, *axis* = `All`, *acceleration* = `None`)

Get/Set target acceleration.

(NOT the current acceleration!)

The default acceleration set on power on is 100 deg/(s∗s).

- If axis is All and acceleration is None then a NUMBER_OF_AXES-list of the currently 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∗second).

10.14.3.2 def sdh.sdhserial.cSDHSerial.alim (*self*, *axis* = `All`)

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.

Acceleration limits are reported in degrees per (second∗second).

10.14.3.3 def sdh.sdhserial.cSDHSerial.AxisCommand (*self*, *command*, *axis* = `All`, *value* = `None`)

Get/Set values.

- If axis is All and value is None then a NUMBER_OF_AXES-list of the current values read from the SDH is returned
- If axis is a single number and value is None then the current 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.

10.14.3.4 def sdh.sdhserial.cSDHSerial.Close (*self*)

Close connection to serial interface.

10.14.3.5 def sdh.sdhserial.cSDHSerial.con (*self*, *controller* = `None`)

Get/set controller type.

If controller is None then the currently set controller is read from the SDH firmware and returned. Else the given controller type is set in the SDH firmware if valid.

10.14.3.6 def sdh.sdhserial.cSDHSerial.debug (*self*, *value*)

10.14.3.7 def sdh.sdhserial.cSDHSerial.demo (*self*, *onoff*)

Enable/disable SCHUNK demo.

10.14.3.8 def sdh.sdhserial.cSDHSerial.ExtractFirmwareState (*self*, *lines*)

Try to extract the state of the SDH firmware from the lines reply.

10.14.3.9 def sdh.sdhserial.cSDHSerial.get_duration (*self*)

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

10.14.3.10 def sdh.sdhserial.cSDHSerial.GetDuration (*self*, *line*)

Return duration of the execution of a SDH command as reported by line.

10.14.3.11 def sdh.sdhserial.cSDHSerial.grip (*self*, *close*, *velocity*, *sequ*)

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

10.14.3.12 def sdh.sdhserial.cSDHSerial.id (*self*)

Return id of SDH.

10.14.3.13 def sdh.sdhserial.cSDHSerial.igrip (*self*, *axis* = `All`, *limit* = `None`)

Get/Set motor current limits for grip commands.

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the currently set current limits is returned

- If axis is a single number and limit is None then the current limit for that axis is returned.
- If axis and limit are single numbers then the current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.14.3.14 def sdh.sdhserial.cSDHSerial.ihold (*self*, *axis* = `All`, *limit* = `None`)

Get/Set motor current limits for hold commands.

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the currently set current limits is returned
- If axis is a single number and limit is None then the current limit for that axis is returned.
- If axis and limit are single numbers then the current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.14.3.15 def sdh.sdhserial.cSDHSerial.ilim (*self*, *axis* = `All`, *limit* = `None`)

Get/Set current limit for m command.

- If axis is All and limit is None then a NUMBER_OF_AXES-list of the currently set current limits is returned
- If axis is a single number and limit is None then the current limit for that axis is returned.
- If axis and limit are single numbers then the current limit for that axis is set (and returned).
- If axis is All and limit is a NUMBER_OF_AXES-vector then all axes current limits are set accordingly, the NUMBER_OF_AXES-list is returned.

10.14.3.16 def sdh.sdhserial.cSDHSerial.kv (*self*, *axis* = `All`, *kv* = `None`)

Get/Set kv parameter.

- If axis is All and kv is None then a NUMBER_OF_AXES-list of the currently 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

10.14.3.17 def sdh.sdhserial.cSDHSerial.m (*self*, *sequ*)

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 currently 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

10.14.3.18 def sdh.sdhserial.cSDHSerial.numaxis (*self*)

Return number of axis of SDH.

10.14.3.19 def sdh.sdhserial.cSDHSerial.p (*self*, *axis* = `All`, *angle* = `None`)

Get/Set target angle for axis.

(NOT the current angle!)

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the currently 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.

10.14.3.20 def sdh.sdhserial.cSDHSerial.p_max (*self*, *axis* = `All`, *angle* = `None`)

Get/Set maximum allowed target angle for axis.

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the currently set maximum angles is returned
- If axis is a single number and angle is None then the maximum angle for that axis is returned.
- If axis and angle are single numbers then the maximum angle for that axis is set (and returned).
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes maximum angles are set accordingly, the NUMBER_OF_AXES-list is returned.
- This will yield a E_RANGE_ERROR if any of the new maximum positions to set is smaller than the actual position or the current minimum position of the axis.

Angles are set/reported in degrees.

10.14.3.21 def sdh.sdhserial.cSDHSerial.p_min (*self*, *axis* = `All`, *angle* = `None`)

Get/Set minimum allowed target angle for axis.

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the currently set minimum angles is returned
- If axis is a single number and angle is None then the minimum angle for that axis is returned.
- If axis and angle are single numbers then the minimum angle for that axis is set (and returned).
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes minimum angles are set accordingly, the NUMBER_OF_AXES-list is returned.
- This will yield a E_RANGE_ERROR if any of the new minimum positions to set is larger than the actual position or the current maximum position of the axis.

Angles are set/reported in degrees.

10.14.3.22 def sdh.sdhserial.cSDHSerial.p_offset (*self*, *axis* = `All`, *angle* = `None`)

Get/Set offset for axis.

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the currently set offset angles is returned

- If axis is a single number and angle is None then the offset angle for that axis is returned.
- If axis and angle are single numbers then the offset angle for that axis is set (and returned).
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes offset angles are set accordingly, the NUMBER_OF_AXES-list is returned.

Angles are set/reported in degrees.

10.14.3.23 def sdh.sdhserial.cSDHSerial.pid (*self*, *axis*, *p* = `None`, *i* = `None`, *d* = `None`)

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 currently 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

10.14.3.24 def sdh.sdhserial.cSDHSerial.pos (*self*, *axis* = `All`)

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.

10.14.3.25 def sdh.sdhserial.cSDHSerial.pos_save (*self*, *axis* = `All`, *value* = `None`)

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

10.14.3.26 def sdh.sdhserial.cSDHSerial.power (*self*, *axis* = `All`, *flag* = `None`)

Get/Set current power state.

- If axis is All and flag is None then a NUMBER_OF_AXES-list of the currently set power states is returned
- If axis is a single number and flag is None then the power state for that axis is returned.
- If axis is a single number and flag is a single number or a boolean value then the power state for that axis is set (and returned).
- If axis is All and flag is a NUMBER_OF_AXES-vector then all axes power states are set accordingly, the NUMBER_OF_AXES-list is returned.
- If axis is All and flag is a a single number or a boolean value then all axes power states are set to that value, the NUMBER_OF_AXES-list is returned.

10.14.3.27 def sdh.sdhserial.cSDHSerial.property (*self*, *propname*, *value*)

Set named property.

Valid propnames are:

- "user_errors"
- "terminal"
- "debug"

10.14.3.28 def sdh.sdhserial.cSDHSerial.ref (*self*, *axis* = `All`, *value* = `None`)

Do reference movements with selected axes.

(Usefull for axes that dont have an absolute encoder)

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 value is None then an exception is thrown since this is NOT usefull here
- 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

10.14.3.29 def sdh.sdhserial.cSDHSerial.rvel (*self*, *axis* = `All`)

Get reference angular velocity/ies of axis/axes.

- If axis is All then a NUMBER_OF_AXES-vector of the actual angular velocity is returned
- If axis is a single number then the actual angular velocity of that axis is returned.

Angular velocities are reported in degrees per second.

10.14.3.30 def sdh.sdhserial.cSDHSerial.selgrip (*self*, *grip*, *sequ*)

Send "selgrip grip" command to SDH.

Where grip is in [0..self.NUMBER_OF_GRIPS-1] or one of the self.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.

return the expected duration of the execution of the command in seconds

10.14.3.31 def sdh.sdhserial.cSDHSerial.Send (*self*, *s*, *nb_lines* = `All`, *nb_lines_total* = `All`)

Send command string s+EOL to self.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.

self.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.

10.14.3.32 def sdh.sdhserial.cSDHSerial.SendParse (*self, s, re_obj*)

Simplified parsing of 1 line commands.

s is the command to send re_obj is a compiled regular expression object the reply for s from the SDH is matched against re_obj and the group 1 of the resulting match object is returned. In case of errors the procedure is repeated up to 3 times after syncing the output

10.14.3.33 def sdh.sdhserial.cSDHSerial.sn (*self*)

Return sn of SDH.

10.14.3.34 def sdh.sdhserial.cSDHSerial.soc (*self*)

Return soc of SDH.

10.14.3.35 def sdh.sdhserial.cSDHSerial.soc_date (*self*)

Return soc of SDH.

10.14.3.36 def sdh.sdhserial.cSDHSerial.state (*self, axis =* `All`)

Get actual state/s of axis/axes.

state values are returned numerically, see eAxisState.

- 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.

10.14.3.37 def sdh.sdhserial.cSDHSerial.stop (*self*)

Stop sdh.

Will NOT interrupt a previous "selgrip" or "grip" command, only an "m" command!

10.14.3.38 def sdh.sdhserial.cSDHSerial.Sync (*self*)

Read all pending lines from SDH to resync execution of PC and SDH.

10.14.3.39 def sdh.sdhserial.cSDHSerial.temp (*self*)

Get actual temperatures of SDH.

Returns a list of the actual controller and driver temperature in degrees celsius.

10.14.3.40 def sdh.sdhserial.cSDHSerial.terminal (*self*, *value*)

10.14.3.41 def sdh.sdhserial.cSDHSerial.tpap (*self*, *axis* = `All`, *angle* = `None`)

Set target angle, get actual angle for axis.

- If axis is All and angle is None then a NUMBER_OF_AXES-list of the currently set target angles is returned
- If axis is a single number and angle is None then the actual angle for that axis is returned.
- If axis and angle are single numbers then the target angle for that axis is set (and actual angle returned).
- If axis is All and angle is a NUMBER_OF_AXES-vector then all axes target angles are set accordingly, the NUMBER_OF_AXES-list of actual angles is returned.

Angles are set/reported in degrees.

10.14.3.42 def sdh.sdhserial.cSDHSerial.tvav (*self*, *axis* = `All`, *velocity* = `None`)

Set target velocity, get actual velocity for axis.

- If axis is All and velocity is None then a NUMBER_OF_AXES-list of the currently set target velocities is returned
- If axis is a single number and velocity is None then the actual velocity for that axis is returned.
- If axis and velocity are single numbers then the target velocity for that axis is set (and actual velocity returned).
- If axis is All and velocity is a NUMBER_OF_AXES-vector then all axes target velocities are set accordingly, the NUMBER_OF_AXES-list of actual velocities is returned.

Angles are set/reported in degrees.

10.14.3.43 def sdh.sdhserial.cSDHSerial.user_errors (*self*, *value*)

10.14.3.44 def sdh.sdhserial.cSDHSerial.v (*self*, *axis* = `All`, *velocity* = `None`)

Get/Set target velocity.

(NOT the current velocity!)

The default velocity set on power on is 40 deg/s.

- If axis is All and velocity is None then a NUMBER_OF_AXES-list of the currently 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.

10.14.3.45 def sdh.sdhserial.cSDHSerial.vel (*self*, *axis* = `All`)

Get actual angular velocity/ies of axis/axes.

- If axis is All then a NUMBER_OF_AXES-vector of the actual angular velocity is returned
- If axis is a single number then the actual angular velocity of that axis is returned.

Angular velocities are reported in degrees per second.

10.14.3.46 def sdh.sdhserial.cSDHSerial.ver (*self*)

Return version of SDH firmware.

10.14.3.47 def sdh.sdhserial.cSDHSerial.ver_date (*self*)

Return date of SDH firmware.

10.14.3.48 def sdh.sdhserial.cSDHSerial.vlim (*self*, *axis* = `All`)

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.

Velocity limits are reported in degrees per second.

10.14.3.49 def sdh.sdhserial.cSDHSerial.vp (*self*, *velocity_profile* = `None`)

Get/set velocity profile.

If velocity_profile is None then the currently 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.

10.14.4 Member Data Documentation

10.14.4.1 sdh.sdhserial.cSDHSerial.actual_con

10.14.4.2 sdh.sdhserial.cSDHSerial.actual_vp

10.14.4.3 sdh.sdhserial.cSDHSerial.com

the RS232 connection to use for communication

10.14.4.4 sdh.sdhserial.cSDHSerial.EOL

String to use as "End Of Line" marker when sending to SDH.

10.14.4.5 sdh.sdhserial.cSDHSerial.firmware_state

the last known state of the SDH firmware

Reimplemented from sdh.sdhbase.cSDHBase.

10.14.4.6 sdh.sdhserial.cSDHSerial.m_sequtime

additional time in seconds to wait for sequential execution of "m"-command (as these are always executed non-sequentially by the SDH firmware) (no longer needed since WaitAxis() is used to ensure movement has ended)

10.14.4.7 sdh.sdhserial.cSDHSerial.nb_lines_to_ignore

number of remaining reply lines of a previous (non sequential) command

Reimplemented from sdh.sdhbase.cSDHBase.

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

- sdh/sdhserial.py

10.15 sdh.tkdsa.cSDHTactileSensorPatch Class Reference

A class to store a tactile sensor patch.

Public Member Functions

- def __init__

  Constructor of cSDHTactileSensorPatch:

- def GetTexel

Public Attributes

- part
- fi
- ts
- m
- columns
- rows
- bit_resolution
- maxvalue

10.15.1 Detailed Description

A class to store a tactile sensor patch.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 def sdh.tkdsa.cSDHTactileSensorPatch.__init__ (*self*, *fi*, *part*, *ts*)

Constructor of cSDHTactileSensorPatch:

10.15.3 Member Function Documentation

10.15.3.1 def sdh.tkdsa.cSDHTactileSensorPatch.GetTexel (*self*, *x*, *y*)

10.15.4 Member Data Documentation

10.15.4.1 sdh.tkdsa.cSDHTactileSensorPatch.bit_resolution

10.15.4.2 sdh.tkdsa.cSDHTactileSensorPatch.columns

10.15.4.3 sdh.tkdsa.cSDHTactileSensorPatch.fi

10.15.4.4 sdh.tkdsa.cSDHTactileSensorPatch.m

10.15.4.5 sdh.tkdsa.cSDHTactileSensorPatch.maxvalue

10.15.4.6 sdh.tkdsa.cSDHTactileSensorPatch.part

10.15.4.7 sdh.tkdsa.cSDHTactileSensorPatch.rows

10.15.4.8 sdh.tkdsa.cSDHTactileSensorPatch.ts

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

- sdh/tkdsa.py

10.16 sdh.auxiliary.cSphere Class Reference

A class to represent sphere objects.

Public Member Functions

- def __init__
  Constructor, store coordinates x,y,z and radius r of spehre.
- def Distance
  Calculates the distance between self and other sphere.
- def Toiv
  Return a string of this sphere as an OpenInventor iv description.

Public Attributes

- z
- r

10.16.1 Detailed Description

A class to represent sphere objects. Used for simple internal collision check.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 def sdh.auxiliary.cSphere.__init__ (*self, x, y, z, r*)

Constructor, store coordinates x,y,z and radius r of spehre.

10.16.3 Member Function Documentation

10.16.3.1 def sdh.auxiliary.cSphere.Distance (*self, other*)

Calculates the distance between self and other sphere.

If the returned result is zero then the spheres touch each other, If it is negative the spheres intersect each other.

10.16.3.2 def sdh.auxiliary.cSphere.Toiv (*self*)

Return a string of this sphere as an OpenInventor iv description.

10.16.4 Member Data Documentation

10.16.4.1 sdh.auxiliary.cSphere.r

10.16.4.2 sdh.auxiliary.cSphere.z

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

- sdh/auxiliary.py

10.17 demo-gui.cTkSDHApplication Class Reference

The "Application" class of the simple SDH GUI.

Public Member Functions

- def __init__

  Constructor of cTkSDHApplication.
- def CreateWidgets

  Create the GUI widgets:
- def ScaleChanged

  One of the scales has moved.
- def QuitAndKeep

  Quit but keep the controllers enabled.
- def SetToSpecific

  Set all axis sliders of all fingers to the values specified in angles.
- def GetSliders

  Get all axis sliders of all fingers.
- def SetToActual

  Set all axis sliders of all fingers to their current actual angle.
- def SetToActualToggle

  Toggle keeping SetToActual.
- def SetToActualKeep

  Call SetToActual periodically if desired.
- def MoveHand

  Set target positons for all fingers from gui and make hand (all fingers) move there.

- def Stop
  Stop movement of fingers (keep controllers enabled)
- def FastStop
  Fast stop movement of fingers (disable controllers)
- def GetTemperature
  GetTemperatures of axis motors, FPGA and PCB.
- def ShowTactileSensors
  Show the tactile sensors.
- def UpdateTSFrame

Public Attributes

- options
- ts_toplevel
- pid_toplevel
- me_menue
- fi_finger0
- fi_finger1
- fi_finger2
- bb_buttons
- keep_actual_button_no
- keep_actual
- l_temperature
- l_logo
- sps_save_poses
- gr_grip

10.17.1 Detailed Description

The "Application" class of the simple SDH GUI.

- creates the widgets
- defines Keyboard shortcuts (see docstring of file)
- defines callbacks to command the SDH

10.17.2 Constructor & Destructor Documentation

10.17.2.1 def demo-gui.cTkSDHApplication.__init__ (*self*, *options* = `None`, *master* = `None`)

Constructor of cTkSDHApplication.

10.17.3 Member Function Documentation

10.17.3.1 def demo-gui.cTkSDHApplication.CreateWidgets (*self*)

Create the GUI widgets:

10.17.3.2 def demo-gui.cTkSDHApplication.FastStop (*self*, *event* = None)

Fast stop movement of fingers (disable controllers)

10.17.3.3 def demo-gui.cTkSDHApplication.GetSliders (*self*, *event* = None)

Get all axis sliders of all fingers.

10.17.3.4 def demo-gui.cTkSDHApplication.GetTemperature (*self*, *event* = None)

GetTemperatures of axis motors, FPGA and PCB.

10.17.3.5 def demo-gui.cTkSDHApplication.MoveHand (*self*, *event* = None)

Set target positons for all fingers from gui and make hand (all fingers) move there.

10.17.3.6 def demo-gui.cTkSDHApplication.QuitAndKeep (*self*, *event*)

Quit but keep the controllers enabled.

10.17.3.7 def demo-gui.cTkSDHApplication.ScaleChanged (*self*, *v*)

One of the scales has moved.

10.17.3.8 def demo-gui.cTkSDHApplication.SetToActual (*self*, *event* = None)

Set all axis sliders of all fingers to their current actual angle.

10.17.3.9 def demo-gui.cTkSDHApplication.SetToActualKeep (*self*)

Call SetToActual periodically if desired.

10.17.3.10 def demo-gui.cTkSDHApplication.SetToActualToggle (*self*, *event* = None, *flag* = None)

Toggle keeping SetToActual.

10.17.3.11 def demo-gui.cTkSDHApplication.SetToSpecific (*self*, *event*, *angles*)

Set all axis sliders of all fingers to the values specified in angles.

10.17.3.12 def demo-gui.cTkSDHApplication.ShowTactileSensors (*self*, *flag*, *style* = `["color"]`)

Show the tactile sensors.

10.17.3.13 def demo-gui.cTkSDHApplication.Stop (*self*, *event* = `None`)

Stop movement of fingers (keep controllers enabled)

10.17.3.14 def demo-gui.cTkSDHApplication.UpdateTSFrame (*self*)

10.17.4 Member Data Documentation

10.17.4.1 demo-gui.cTkSDHApplication.bb_buttons

10.17.4.2 demo-gui.cTkSDHApplication.fi_finger0

10.17.4.3 demo-gui.cTkSDHApplication.fi_finger1

10.17.4.4 demo-gui.cTkSDHApplication.fi_finger2

10.17.4.5 demo-gui.cTkSDHApplication.gr_grip

10.17.4.6 demo-gui.cTkSDHApplication.keep_actual

10.17.4.7 demo-gui.cTkSDHApplication.keep_actual_button_no

10.17.4.8 demo-gui.cTkSDHApplication.l_logo

10.17.4.9 demo-gui.cTkSDHApplication.l_temperature

10.17.4.10 demo-gui.cTkSDHApplication.me_menue

10.17.4.11 demo-gui.cTkSDHApplication.options

10.17.4.12 demo-gui.cTkSDHApplication.pid_toplevel

10.17.4.13 demo-gui.cTkSDHApplication.sps_save_poses

10.17.4.14 demo-gui.cTkSDHApplication.ts_toplevel

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

- demo/demo-gui.py

10.18 demo-gui.cTkSDHButtonBox Class Reference

A simple box for buttons.

Public Member Functions

- def __init__

  Constructor of cTkSDHButtonBox.

- def AddButton

  Add a button to the button box, layout int auto-grid horizontally.

Public Attributes

- buttons
- nb_buttons

10.18.1 Detailed Description

A simple box for buttons.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 def demo-gui.cTkSDHButtonBox.__init__ (*self*, *master* = None)

Constructor of cTkSDHButtonBox.

10.18.3 Member Function Documentation

10.18.3.1 def demo-gui.cTkSDHButtonBox.AddButton (*self*, *text* = "", *command* = None, *underline* = None, *ipadx* = None, *ipady* = None, *padx* = None, *pady* = None, *bg* = None, *fg* = None)

Add a button to the button box, layout int auto-grid horizontally.

10.18.4 Member Data Documentation

10.18.4.1 demo-gui.cTkSDHButtonBox.buttons

10.18.4.2 demo-gui.cTkSDHButtonBox.nb_buttons

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

- demo/demo-gui.py

10.19 demo-gui.cTkSDHCurrent Class Reference

Toplevel window to show and adjust the motor current parameters of an SDH.

Public Member Functions

- def __init__
- def CheckValues
- def ReturnPressed
- def UpdateToSDHTemporarily

  Update the SDH current parameters from the entries in the cTkSDHCurrent toplevel window The values are stored temporarily, i.e.

- def UpdateFromSDH

  Update the entries in the cTkSDHCurrent toplevel window from the SDH.

Public Attributes

- tl
- currents

10.19.1 Detailed Description

Toplevel window to show and adjust the motor current parameters of an SDH.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 def demo-gui.cTkSDHCurrent.__init__ (*self*, *master* = None)

10.19.3 Member Function Documentation

10.19.3.1 def demo-gui.cTkSDHCurrent.CheckValues (*self*)

10.19.3.2 def demo-gui.cTkSDHCurrent.ReturnPressed (*self*, *event*)

10.19.3.3 def demo-gui.cTkSDHCurrent.UpdateFromSDH (*self*)

Update the entries in the cTkSDHCurrent toplevel window from the SDH.

10.19.3.4 def demo-gui.cTkSDHCurrent.UpdateToSDHTemporarily (*self*)

Update the SDH current parameters from the entries in the cTkSDHCurrent toplevel window The values are stored temporarily, i.e.

remain active until changed or until power cycle or reset

10.19.4 Member Data Documentation

10.19.4.1 demo-gui.cTkSDHCurrent.currents

10.19.4.2 demo-gui.cTkSDHCurrent.tl

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

- demo/demo-gui.py

10.20 demo-gui.cTkSDHFinger Class Reference

A widget for a single finger.

Public Member Functions

- def __init__

  Constructor of cTkSDHFinger: create a widget to control finger with index iFinger.

- def CreateWidgets

  Create GUI elements for one finger.

- def ShowCollision

  Show collision state.

- def SetToActual

  Update GUI elements (sliders) from the actual position of the fingers axes.

- def SetAsTarget

  Set target positions for finger self.iFinger.

- def MoveFinger

  Move finger self.iFinger to the target positions set by the sliders.

Public Attributes

- iFinger
- l_title
- sc_axis_distal
- sc_axis_proximal
- sc_axis_base
- bt_move

10.20.1 Detailed Description

A widget for a single finger.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 def demo-gui.cTkSDHFinger.__init__ (*self, iFinger, master* = `None`)

Constructor of cTkSDHFinger: create a widget to control finger with index *iFinger*.

10.20.3 Member Function Documentation

10.20.3.1 def demo-gui.cTkSDHFinger.CreateWidgets (*self*)

Create GUI elements for one finger.

10.20.3.2 def demo-gui.cTkSDHFinger.MoveFinger (*self, event* = `None`)

Move finger self.iFinger to the target positions set by the sliders.

10.20.3.3 def demo-gui.cTkSDHFinger.SetAsTarget (*self*)

Set target positions for finger self.iFinger.

10.20.3.4 def demo-gui.cTkSDHFinger.SetToActual (*self*)

Update GUI elements (sliders) from the actual position of the fingers axes.

10.20.3.5 def demo-gui.cTkSDHFinger.ShowCollision (*self*, *collision*, *dist*)

Show collision state.

10.20.4 Member Data Documentation

10.20.4.1 demo-gui.cTkSDHFinger.bt_move

10.20.4.2 demo-gui.cTkSDHFinger.iFinger

10.20.4.3 demo-gui.cTkSDHFinger.l_title

10.20.4.4 demo-gui.cTkSDHFinger.sc_axis_base

10.20.4.5 demo-gui.cTkSDHFinger.sc_axis_distal

10.20.4.6 demo-gui.cTkSDHFinger.sc_axis_proximal

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

- demo/demo-gui.py

10.21 demo-gui.cTkSDHGrip Class Reference

A widget to access the grip skills stored in the SDH.

Public Member Functions

- def __init__

  Constructor of cTkSDHGrip.

- def CreateWidgets

  Create GUI elements.

- def PerformGrip

  Perform grip.

Public Attributes

- iv_gripno
- sc_close
- sc_velocity
- bb_buttons

10.21.1 Detailed Description

A widget to access the grip skills stored in the SDH.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 def demo-gui.cTkSDHGrip.__init__ (*self*, *master* = None)

Constructor of cTkSDHGrip.

10.21.3 Member Function Documentation

10.21.3.1 def demo-gui.cTkSDHGrip.CreateWidgets (*self*)

Create GUI elements.

10.21.3.2 def demo-gui.cTkSDHGrip.PerformGrip (*self*)

Perform grip.

10.21.4 Member Data Documentation

10.21.4.1 demo-gui.cTkSDHGrip.bb_buttons

10.21.4.2 demo-gui.cTkSDHGrip.iv_gripno

10.21.4.3 demo-gui.cTkSDHGrip.sc_close

10.21.4.4 demo-gui.cTkSDHGrip.sc_velocity

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

- demo/demo-gui.py

10.22 miniterm.cTkSDHInterfaceSelectorFrame Class Reference

A toplevel widget class, used to interactively select the communication interface of the miniterm.py app on start.

Public Member Functions

- def __init__

    Constructor of cTkSDHInterfaceSelectorFrame.

- def RS232Callback
- def CANCallback
- def OKCallback
- def CreateWidgets

    Create the GUI widgets:

Public Attributes

- p
- available_ports

10.22.1 Detailed Description

A toplevel widget class, used to interactively select the communication interface of the miniterm.py app on start.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 def miniterm.cTkSDHInterfaceSelectorFrame.__init__ (*self*, *master* = `None`)

Constructor of cTkSDHInterfaceSelectorFrame.

10.22.3 Member Function Documentation

10.22.3.1 def miniterm.cTkSDHInterfaceSelectorFrame.CANCallback (*self*)

10.22.3.2 def miniterm.cTkSDHInterfaceSelectorFrame.CreateWidgets (*self*)

Create the GUI widgets:

10.22.3.3 def miniterm.cTkSDHInterfaceSelectorFrame.OKCallback (*self*)

10.22.3.4 def miniterm.cTkSDHInterfaceSelectorFrame.RS232Callback (*self*)

10.22.4 Member Data Documentation

10.22.4.1 miniterm.cTkSDHInterfaceSelectorFrame.available_ports

10.22.4.2 miniterm.cTkSDHInterfaceSelectorFrame.p

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

- demo/miniterm.py

10.23 demo-gui.cTkSDHInterfaceSelectorToplevel Class Reference

A toplevel widget class, used to select the communication interface to the SDH.

Public Member Functions

- def __init__

    Constructor of cTkSDHInterfaceSelectorToplevel.

- def CBQuit
- def RS232Callback
- def CANCallback
- def OKCallback
- def CreateWidgets

    Create the GUI widgets:

Public Attributes

- options
- parser
- p
- available_ports
- tcp_adr

10.23.1 Detailed Description

A toplevel widget class, used to select the communication interface to the SDH.

- creates the widgets

10.23.2 Constructor & Destructor Documentation

10.23.2.1 def demo-gui.cTkSDHInterfaceSelectorToplevel.__init__ (*self*, *options*, *parser*, *master* = None)

Constructor of cTkSDHInterfaceSelectorToplevel.

10.23.3 Member Function Documentation

10.23.3.1 def demo-gui.cTkSDHInterfaceSelectorToplevel.CANCallback (*self*)

10.23.3.2 def demo-gui.cTkSDHInterfaceSelectorToplevel.CBQuit (*self*)

10.23.3.3 def demo-gui.cTkSDHInterfaceSelectorToplevel.CreateWidgets (*self*)

Create the GUI widgets:

10.23.3.4 def demo-gui.cTkSDHInterfaceSelectorToplevel.OKCallback (*self*)

10.23.3.5 def demo-gui.cTkSDHInterfaceSelectorToplevel.RS232Callback (*self*)

10.23.4 Member Data Documentation

10.23.4.1 demo-gui.cTkSDHInterfaceSelectorToplevel.available_ports

10.23.4.2 demo-gui.cTkSDHInterfaceSelectorToplevel.options

10.23.4.3 demo-gui.cTkSDHInterfaceSelectorToplevel.p

10.23.4.4 demo-gui.cTkSDHInterfaceSelectorToplevel.parser

10.23.4.5 demo-gui.cTkSDHInterfaceSelectorToplevel.tcp_adr

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

- demo/demo-gui.py

10.24 demo-gui.cTkSDHMenu Class Reference

The Menu for the application.

Public Member Functions

- def __init__
  Constructor of cTkSDHMenu.

- def CreateWidgets

  Create the GUI widgets:

- def CBMenuShowSDHVersionInfo
- def CBMenuShowPIDAdjust
- def CBMenuShowCurrentAdjust
- def CBMenuRef
- def CBMenuRef1p
- def CBMenuRef1m
- def CBMenuRef2p
- def CBMenuRef2m
- def CBMenuRef3p
- def CBMenuRef3m
- def CBMenuRef4p
- def CBMenuRef4m
- def CBMenuRef5p
- def CBMenuRef5m
- def CBMenuRef6p
- def CBMenuRef6m
- def CBMenuRef1s
- def CBMenuRef2s
- def CBMenuRef3s
- def CBMenuRef4s
- def CBMenuRef5s
- def CBMenuRef6s
- def CBMenuPort

  Callback for menu entries in sdh port menu.

- def CBMenuDSAPort

  Callback for menu entries in dsa port menu.

- def CBMenuDebug

  Callback for menu entries in debug menu.

- def CBMenuUnitSystems

  Callback for menu entries in unit systems menu.

- def CBMenuVelocityProfile
- def CBMenuTS

  Callback for menu entries in tactile sensor menu.

Public Attributes

- iv_port
- iv_dsaport
- iv_uc_angle
- iv_uc_angular_velocity
- iv_uc_temperature
- uc_angle_list
- uc_angular_velocity_list
- uc_temperature_list
- dbg_verbosity_list
- iv_velocity_profile
- iv_ts
- iv_velocity_profile_list
- mb_dsaports
- mb_ucs
- mb_dbgs
- mb_vps
- mb_ts
- iv_ts_styles

  !!! from cmd line option?

- mb_ref
- ref_menue_entries
- tl_showpidadjust
- tl_showcurrentadjust

10.24.1 Detailed Description

The Menu for the application.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 def demo-gui.cTkSDHMenu.__init__ (*self*, *master* = `None`)

Constructor of cTkSDHMenu.

10.24.3 Member Function Documentation

10.24.3.1 def demo-gui.cTkSDHMenu.CBMenuDebug (*self*, *a*, *b*, *c*)

Callback for menu entries in debug menu.

10.24.3.2 def demo-gui.cTkSDHMenu.CBMenuDSAPort (*self*, *a*, *b*, *c*)

Callback for menu entries in dsa port menu.

10.24.3.3 def demo-gui.cTkSDHMenu.CBMenuPort (*self*, *a*, *b*, *c*)

Callback for menu entries in sdh port menu.

10.24.3.4 def demo-gui.cTkSDHMenu.CBMenuRef (*self*, *axis*, *direction*)

10.24.3.5 def demo-gui.cTkSDHMenu.CBMenuRef1m (*self*, *a*, *b*, *c*)

10.24.3.6 def demo-gui.cTkSDHMenu.CBMenuRef1p (*self*, *a*, *b*, *c*)

10.24.3.7 def demo-gui.cTkSDHMenu.CBMenuRef1s (*self*, *a*, *b*, *c*)

10.24.3.8 def demo-gui.cTkSDHMenu.CBMenuRef2m (*self*, *a*, *b*, *c*)

10.24.3.9 def demo-gui.cTkSDHMenu.CBMenuRef2p (*self*, *a*, *b*, *c*)

10.24.3.10 def demo-gui.cTkSDHMenu.CBMenuRef2s (*self*, *a*, *b*, *c*)

10.24.3.11 def demo-gui.cTkSDHMenu.CBMenuRef3m (*self*, *a*, *b*, *c*)

10.24.3.12 def demo-gui.cTkSDHMenu.CBMenuRef3p (*self*, *a*, *b*, *c*)

10.24.3.13 def demo-gui.cTkSDHMenu.CBMenuRef3s (*self*, *a*, *b*, *c*)

10.24.3.14 def demo-gui.cTkSDHMenu.CBMenuRef4m (*self*, *a*, *b*, *c*)

10.24.3.15 def demo-gui.cTkSDHMenu.CBMenuRef4p (*self*, *a*, *b*, *c*)

10.24.3.16 def demo-gui.cTkSDHMenu.CBMenuRef4s (*self*, *a*, *b*, *c*)

10.24.3.17 def demo-gui.cTkSDHMenu.CBMenuRef5m (*self*, *a*, *b*, *c*)

10.24.3.18 def demo-gui.cTkSDHMenu.CBMenuRef5p (*self*, *a*, *b*, *c*)

10.24.3.19 def demo-gui.cTkSDHMenu.CBMenuRef5s (*self*, *a*, *b*, *c*)

10.24.3.20 def demo-gui.cTkSDHMenu.CBMenuRef6m (*self*, *a*, *b*, *c*)

10.24.3.21 def demo-gui.cTkSDHMenu.CBMenuRef6p (*self*, *a*, *b*, *c*)

10.24.3.22 def demo-gui.cTkSDHMenu.CBMenuRef6s (*self*, *a*, *b*, *c*)

10.24.3.23 def demo-gui.cTkSDHMenu.CBMenuShowCurrentAdjust (*self*)

10.24.3.24 def demo-gui.cTkSDHMenu.CBMenuShowPIDAdjust (*self*)

10.24.3.25 def demo-gui.cTkSDHMenu.CBMenuShowSDHVersionInfo (*self*)

10.24.3.26 def demo-gui.cTkSDHMenu.CBMenuTS (*self*, *a*, *b*, *c*)

Callback for menu entries in tactile sensor menu.

10.24.3.27 def demo-gui.cTkSDHMenu.CBMenuUnitSystems (*self*, *a*, *b*, *c*)

Callback for menu entries in unit systems menu.

10.24.3.28 def demo-gui.cTkSDHMenu.CBMenuVelocityProfile (*self*, *a*, *b*, *c*)

10.24.3.29 def demo-gui.cTkSDHMenu.CreateWidgets (*self*)

Create the GUI widgets:

10.24.4 Member Data Documentation

10.24.4.1 demo-gui.cTkSDHMenu.dbg_verbosity_list

10.24.4.2 demo-gui.cTkSDHMenu.iv_dsaport

10.24.4.3 demo-gui.cTkSDHMenu.iv_port

10.24.4.4 demo-gui.cTkSDHMenu.iv_ts

10.24.4.5 demo-gui.cTkSDHMenu.iv_ts_styles

!!! from cmd line option?

10.24.4.6 demo-gui.cTkSDHMenu.iv_uc_angle

10.24.4.7 demo-gui.cTkSDHMenu.iv_uc_angular_velocity

10.24.4.8 demo-gui.cTkSDHMenu.iv_uc_temperature

10.24.4.9 demo-gui.cTkSDHMenu.iv_velocity_profile

10.24.4.10 demo-gui.cTkSDHMenu.iv_velocity_profile_list

10.24.4.11 demo-gui.cTkSDHMenu.mb_dbgs

10.24.4.12 demo-gui.cTkSDHMenu.mb_dsaports

10.24.4.13 demo-gui.cTkSDHMenu.mb_ref

10.24.4.14 demo-gui.cTkSDHMenu.mb_ts

10.24.4.15 demo-gui.cTkSDHMenu.mb_ucs

10.24.4.16 demo-gui.cTkSDHMenu.mb_vps

10.24.4.17 demo-gui.cTkSDHMenu.ref_menue_entries

10.24.4.18 demo-gui.cTkSDHMenu.tl_showcurrentadjust

10.24.4.19 demo-gui.cTkSDHMenu.tl_showpidadjust

10.24.4.20 demo-gui.cTkSDHMenu.uc_angle_list

10.24.4.21 demo-gui.cTkSDHMenu.uc_angular_velocity_list

10.24.4.22 demo-gui.cTkSDHMenu.uc_temperature_list

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

- demo/demo-gui.py

10.25 demo-gui.cTkSDHPID Class Reference

Toplevel window to show and adjust the pid parameters of an SDH.

Public Member Functions

- def __init__
- def CheckValues

- def ReturnPressed
- def ControlReturnPressed
- def UpdateToSDHTemporarily

    Update the SDH pid parameters from the entries in the cTkSDHPID toplevel window The values are stored temporarily, i.e.

- def UpdateToSDHPersistently

    Update the SDH pid parameters from the entries in the cTkSDHPID toplevel window The values are stored persistently, i.e.

- def UpdateFromSDH

    Update the entries in the cTkSDHPID toplevel window from the SDH.

Public Attributes

- tl
- p
- i
- d

10.25.1 Detailed Description

Toplevel window to show and adjust the pid parameters of an SDH.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 def demo-gui.cTkSDHPID.__init__ (*self*, *master* = `None`)

10.25.3 Member Function Documentation

10.25.3.1 def demo-gui.cTkSDHPID.CheckValues (*self*)

10.25.3.2 def demo-gui.cTkSDHPID.ControlReturnPressed (*self*, *event*)

10.25.3.3 def demo-gui.cTkSDHPID.ReturnPressed (*self*, *event*)

10.25.3.4 def demo-gui.cTkSDHPID.UpdateFromSDH (*self*)

Update the entries in the cTkSDHPID toplevel window from the SDH.

10.25.3.5 def demo-gui.cTkSDHPID.UpdateToSDHPersistently (*self*)

Update the SDH pid parameters from the entries in the cTkSDHPID toplevel window The values are stored persistently, i.e.

will survive a power cycle or reset

10.25.3.6 def demo-gui.cTkSDHPID.UpdateToSDHTemporarily (*self*)

Update the SDH pid parameters from the entries in the cTkSDHPID toplevel window The values are stored temporarily, i.e.

remain active until changed or until power cycle or reset

10.25.4 Member Data Documentation

10.25.4.1 demo-gui.cTkSDHPID.d

10.25.4.2 demo-gui.cTkSDHPID.i

10.25.4.3 demo-gui.cTkSDHPID.p

10.25.4.4 demo-gui.cTkSDHPID.tl

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

- demo/demo-gui.py

10.26 demo-gui.cTkSDHSavePose Class Reference

A widget to save restore a single pose.

Public Member Functions

- def __init__

  Constructor of cTkSDHSavePose.

- def CreateWidgets

  Create GUI elements for one pose.

- def Validate

  Check if the text in the entry is a valid pose.

- def PoseInput

  Callback for all Keyboard input in the pose entry: accept only numbers, ".", ",", ...

- def SetPose

  Set finger sliders to value from the pose entry.

- def SetPoseMove

  Set finger sliders to value from the pose entry and move there.

- def GetPose

  Get pose entry from the finger sliders.

Public Attributes

- shortcut_set
- bt_set
- bt_get
- en_name
- en_pose
- iv_selected
- cb_selected

10.26.1 Detailed Description

A widget to save restore a single pose.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 def demo-gui.cTkSDHSavePose.__init__ (*self*, *shortcut_set* = "", *name* = "", *posestr* = "", *master* = None)

Constructor of cTkSDHSavePose.

10.26.3 Member Function Documentation

10.26.3.1 def demo-gui.cTkSDHSavePose.CreateWidgets (*self*)

Create GUI elements for one pose.

10.26.3.2 def demo-gui.cTkSDHSavePose.GetPose (*self*, *event* = None)

Get pose entry from the finger sliders.

10.26.3.3 def demo-gui.cTkSDHSavePose.PoseInput (*self*, *event*)

Callback for all Keyboard input in the pose entry: accept only numbers, ".", ",", ... edit keys

10.26.3.4 def demo-gui.cTkSDHSavePose.SetPose (*self*, *event* = None)

Set finger sliders to value from the pose entry.

10.26.3.5 def demo-gui.cTkSDHSavePose.SetPoseMove (*self*, *event* = None)

Set finger sliders to value from the pose entry and move there.

10.26.3.6 def demo-gui.cTkSDHSavePose.Validate (*self*, *event* = None)

Check if the text in the entry is a valid pose.

10.26.4 Member Data Documentation

10.26.4.1 demo-gui.cTkSDHSavePose.bt_get

10.26.4.2 demo-gui.cTkSDHSavePose.bt_set

10.26.4.3 demo-gui.cTkSDHSavePose.cb_selected

10.26.4.4 demo-gui.cTkSDHSavePose.en_name

10.26.4.5 demo-gui.cTkSDHSavePose.en_pose

10.26.4.6 demo-gui.cTkSDHSavePose.iv_selected

10.26.4.7 demo-gui.cTkSDHSavePose.shortcut_set

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

- demo/demo-gui.py

10.27 demo-gui.cTkSDHSavePoses Class Reference

A widget to save/restore all poses.

Public Member Functions

- def __init__

  Constructor of cTkSDHSavePoses.

- def CreateWidgets

  Create GUI elements for all poses + save/load buttons.

- def SaveToFile

  Save the poses to a file.

- def LoadFromFile

Load poses from file, ask for filename from user if not given.

- def LoopOverSelected

  Start/Stop looping over selected poses Looping status is toggled if looping is None or set to looping if True/False.

- def CBLooping

  Acual looping over selected poses callback.

Public Attributes

- filetypes
- initialdir
- l_title
- sp_save_pose
- bb_buttons
- looping
- loop_index

10.27.1 Detailed Description

A widget to save/restore all poses.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 def demo-gui.cTkSDHSavePoses.__init__ (*self*, *master* = `None`)

Constructor of cTkSDHSavePoses.

10.27.3 Member Function Documentation

10.27.3.1 def demo-gui.cTkSDHSavePoses.CBLooping (*self*)

Acual looping over selected poses callback.

10.27.3.2 def demo-gui.cTkSDHSavePoses.CreateWidgets (*self*)

Create GUI elements for all poses + save/load buttons.

10.27.3.3 def demo-gui.cTkSDHSavePoses.LoadFromFile (*self*, *filename* = "")

Load poses from file, ask for filename from user if not given.

The poses are converted from internal to current external unit system.

10.27.3.4 def demo-gui.cTkSDHSavePoses.LoopOverSelected (*self*, *looping* = None)

Start/Stop looping over selected poses Looping status is toggled if looping is None or set to looping if True/False.

10.27.3.5 def demo-gui.cTkSDHSavePoses.SaveToFile (*self*, *filename* = "")

Save the poses to a file.

Ask for filename from user if not given. The poses are always saved in the internal unit system.

10.27.4 Member Data Documentation

10.27.4.1 demo-gui.cTkSDHSavePoses.bb_buttons

10.27.4.2 demo-gui.cTkSDHSavePoses.filetypes

10.27.4.3 demo-gui.cTkSDHSavePoses.initialdir

10.27.4.4 demo-gui.cTkSDHSavePoses.l_title

10.27.4.5 demo-gui.cTkSDHSavePoses.loop_index

10.27.4.6 demo-gui.cTkSDHSavePoses.looping

10.27.4.7 demo-gui.cTkSDHSavePoses.sp_save_pose

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

- demo/demo-gui.py

10.28 demo-tactile.cTkSDHTactileApplication Class Reference

The "Application" class of demo-tactile.py, the simple SDH tactile visualizer.

Public Member Functions

- def __init__

  Constructor of cTkSDHTactileApplication.

- def UpdateTSFrame
- def CreateWidgets

  Create the GUI widgets:

- def Repaint

Public Attributes

- framerate
- ts
- debug_level
- style
- tsps

10.28.1 Detailed Description

The "Application" class of demo-tactile.py, the simple SDH tactile visualizer.

- creates the widgets
- defines Keyboard shortcuts (see docstring of file)
- defines callbacks to command the SDH

10.28.2 Constructor & Destructor Documentation

10.28.2.1 def demo-tactile.cTkSDHTactileApplication.__init__ (*self*, *ts*, *framerate* = `10`, *master* = `None`, *debug_level* = `0`, *style* = `["color"]`)

Constructor of cTkSDHTactileApplication.

Parameters

self	- the instance of the class that this function operates on (the "object")
ts	- the initialized cDSA object to communicate with the remote DSACON32m controller
framerate	- the framerate for updating the GUI (the tactile sensors will always send at maximum speed of 30 FPS)
master	- the master widget of this
debug_level	- level of debug messages to print • 0 = no messages • 1 = print messages of this object • 2 = print messages of underlying tkdsa.cTkSDHTactileSensorPatches as well
style	- style for displaying tactile sensor data, see online help or tkdsa.cTkSDHTactileSensorPatches for available styles

10.28.3 Member Function Documentation

10.28.3.1 def demo-tactile.cTkSDHTactileApplication.CreateWidgets (*self*)

Create the GUI widgets:

10.28.3.2 def demo-tactile.cTkSDHTactileApplication.Repaint (*self*)

10.28.3.3 def demo-tactile.cTkSDHTactileApplication.UpdateTSFrame (*self*)

10.28.4 Member Data Documentation

10.28.4.1 demo-tactile.cTkSDHTactileApplication.debug_level

10.28.4.2 demo-tactile.cTkSDHTactileApplication.framerate

10.28.4.3 demo-tactile.cTkSDHTactileApplication.style

10.28.4.4 demo-tactile.cTkSDHTactileApplication.ts

10.28.4.5 demo-tactile.cTkSDHTactileApplication.tsps

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

- demo/demo-tactile.py

10.29 sdh.tkdsa.cTkSDHTactileSensorPatch Class Reference

A widget to display a single tactile sensor patch.

Public Member Functions

- def __init__

  Constructor of cTkSDHTactileSensorPatch.

- def CreateWidgets

  Create GUI elements for one tactile sensor patch.

- def Repaint
- def ToColor

  Return a pair of background and foreground color for texel at (r,c)

- def ToGrey

  Return a pair of background and foreground color for texel at (r,c)

Public Attributes

- patch
- texel_display_style
- texel

10.29.1 Detailed Description

A widget to display a single tactile sensor patch.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 def sdh.tkdsa.cTkSDHTactileSensorPatch.__init__ (*self*, *patch*, *master* = `None`, *style* = `["color"]`)

Constructor of cTkSDHTactileSensorPatch.

10.29.3 Member Function Documentation

10.29.3.1 def sdh.tkdsa.cTkSDHTactileSensorPatch.CreateWidgets (*self*)

Create GUI elements for one tactile sensor patch.

10.29.3.2 def sdh.tkdsa.cTkSDHTactileSensorPatch.Repaint (*self*)

10.29.3.3 def sdh.tkdsa.cTkSDHTactileSensorPatch.ToColor (*self*, *r*, *c*)

Return a pair of background and foreground color for texel at (r,c)

10.29.3.4 def sdh.tkdsa.cTkSDHTactileSensorPatch.ToGrey (*self*, *r*, *c*)

Return a pair of background and foreground color for texel at (r,c)

10.29.4 Member Data Documentation

10.29.4.1 sdh.tkdsa.cTkSDHTactileSensorPatch.patch

10.29.4.2 sdh.tkdsa.cTkSDHTactileSensorPatch.texel

10.29.4.3 sdh.tkdsa.cTkSDHTactileSensorPatch.texel_display_style

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

- sdh/tkdsa.py

10.30 sdh.tkdsa.cTkSDHTactileSensorPatches Class Reference

Widget to display all tactile sensor patches of an SDH.

Public Member Functions

- def __init__

  Constructor of cTkSDHTactileSensorPatches.

- def CreateWidgets

  Create the GUI widgets:

- def Repaint

Public Attributes

- debug_level
- ts
- tsps

10.30.1 Detailed Description

Widget to display all tactile sensor patches of an SDH.

- creates the widgets
- defines Keyboard shortcuts (see docstring of file)
- defines callbacks to command the SDH

10.30.2 Constructor & Destructor Documentation

10.30.2.1 def sdh.tkdsa.cTkSDHTactileSensorPatches.__init__ (*self*, *ts*, *master* = `None`, *debug_level* = `0`, *debug_output* = `sys.stderr`, *style* = `["color"]`)

Constructor of cTkSDHTactileSensorPatches.

10.30.3 Member Function Documentation

10.30.3.1 def sdh.tkdsa.cTkSDHTactileSensorPatches.CreateWidgets (*self*, *style*)

Create the GUI widgets:

10.30.3.2 def sdh.tkdsa.cTkSDHTactileSensorPatches.Repaint (*self*)

10.30.4 Member Data Documentation

10.30.4.1 sdh.tkdsa.cTkSDHTactileSensorPatches.debug_level

10.30.4.2 sdh.tkdsa.cTkSDHTactileSensorPatches.ts

10.30.4.3 sdh.tkdsa.cTkSDHTactileSensorPatches.tsps

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

- sdh/tkdsa.py

10.31 sdh.unit.cUnitConverter Class Reference

Unit conversion class.

Public Member Functions

- def __init__
  Constructor of cUnitConverter class.
- def ToExternal
  Convert value 'internal' given in internal 'self.name' units into external units.
- def ToInternal
  Convert value 'external' given in external 'self.name' units into internal units.

Public Attributes

- kind
  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.

10.31.1 Detailed Description

Unit conversion class. 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.

Unit conversion class. See html/pdf documentation for details.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 def sdh.unit.cUnitConverter.__init__ (*self*, *kind*, *name*, *symbol*, *factor* = `1.0`, *offset* = `0.0`, *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

self	- reference to the object itself
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

Constructor of cUnitConverter class.

10.31.3 Member Function Documentation

10.31.3.1 def sdh.unit.cUnitConverter.ToExternal (*self*, *internal*)

Convert value 'internal' given in internal 'self.name' units into external units.

Returns internal $*$ factor + offset

The value 'internal' can be a single number or a vector-like object (list, tuple, array.array). In the latter 3 cases every member of the vector is converted and a new object of the same type is returned

10.31.3.2 def sdh.unit.cUnitConverter.ToInternal (*self*, *external*)

Convert value 'external' given in external 'self.name' units into internal units.

Returns (external - offset) / factor

The value 'external' can be a single number or a vector-like object (list, tuple, array.array). In the latter 3 cases every member of the vector is converted and a new object of the same type is returned

10.31.4 Member Data Documentation

10.31.4.1 sdh.unit.cUnitConverter.decimal_places

A usefull number of decimal places for printing values in the external unit system.

10.31.4.2 sdh.unit.cUnitConverter.factor

the conversion factor from internal to external units

10.31.4.3 sdh.unit.cUnitConverter.kind

the kind of unit to be converted (something like "angle" or "time")

10.31.4.4 sdh.unit.cUnitConverter.name

the name of the external unit (something like "degrees" or "milliseconds")

10.31.4.5 sdh.unit.cUnitConverter.offset

the conversion offset from internal to external units

10.31.4.6 sdh.unit.cUnitConverter.symbol

the symbol of the external unit (something like "deg" or "ms")

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

- sdh/unit.py

10.32 sdh.utils.DefaultDict Class Reference

Dictionary with a default value for unknown keys.

Public Member Functions

- def __init__
- def __getitem__
- def __copy__

Public Attributes

- default

10.32.1 Detailed Description

Dictionary with a default value for unknown keys.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 def sdh.utils.DefaultDict.__init__ (*self*, *default*)

10.32.3 Member Function Documentation

10.32.3.1 def sdh.utils.DefaultDict.__copy__ (*self*)

10.32.3.2 def sdh.utils.DefaultDict.__getitem__ (*self*, *key*)

10.32.4 Member Data Documentation

10.32.4.1 sdh.utils.DefaultDict.default

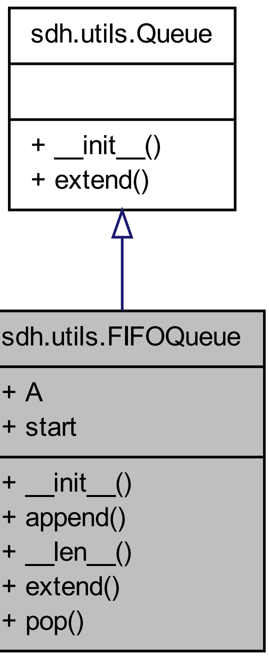
The documentation for this class was generated from the following file:

- sdh/utils.py

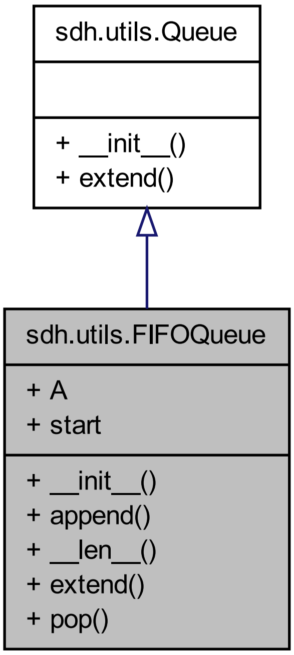
10.33 sdh.utils.FIFOQueue Class Reference

A First-In-First-Out Queue.

Inheritance diagram for sdh.utils.FIFOQueue:



Collaboration diagram for sdh.utils.FIFOQueue:



Public Member Functions

- def __init__
- def append
- def __len__
- def extend
- def pop

Public Attributes

- A
- start

10.33.1 Detailed Description

A First-In-First-Out Queue.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 def sdh.utils.FIFOQueue.__init__ (*self*)

Reimplemented from sdh.utils.Queue.

10.33.3 Member Function Documentation

10.33.3.1 def sdh.utils.FIFOQueue.__len__ (*self*)

10.33.3.2 def sdh.utils.FIFOQueue.append (*self*, *item*)

10.33.3.3 def sdh.utils.FIFOQueue.extend (*self*, *items*)

Reimplemented from sdh.utils.Queue.

10.33.3.4 def sdh.utils.FIFOQueue.pop (*self*)

10.33.4 Member Data Documentation

10.33.4.1 sdh.utils.FIFOQueue.A

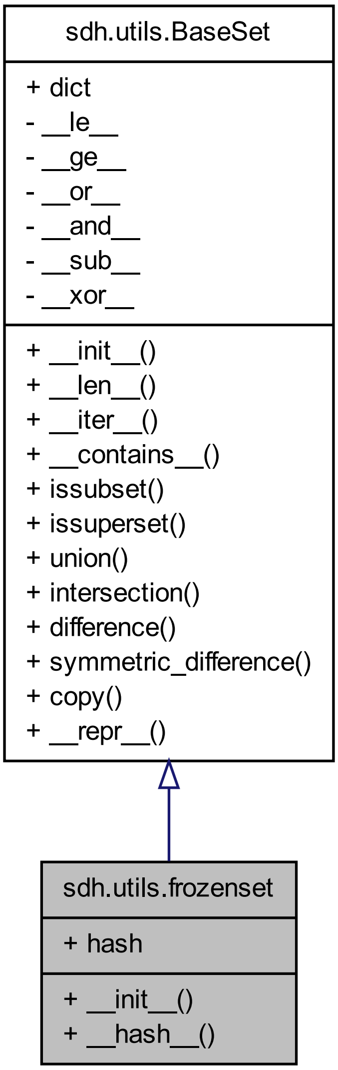
10.33.4.2 sdh.utils.FIFOQueue.start

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

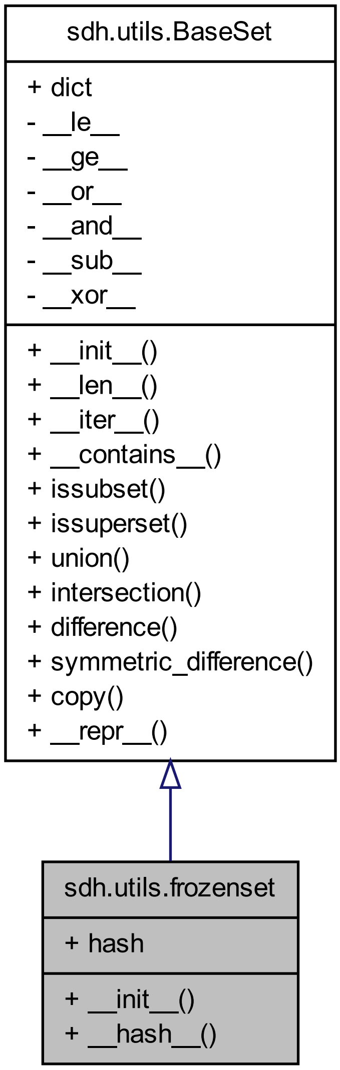
- sdh/utils.py

10.34 sdh.utils.frozenset Class Reference

Inheritance diagram for sdh.utils.frozenset:



Collaboration diagram for sdh.utils.frozenset:



Public Member Functions

- def __init__
- def __hash__

Public Attributes

- hash

10.34.1 Constructor & Destructor Documentation

10.34.1.1 def sdh.utils.frozenset.__init__ (*self*, *elements* = [])

Reimplemented from sdh.utils.BaseSet.

10.34.2 Member Function Documentation

10.34.2.1 def sdh.utils.frozenset.__hash__ (*self*)

10.34.3 Member Data Documentation

10.34.3.1 sdh.utils.frozenset.hash

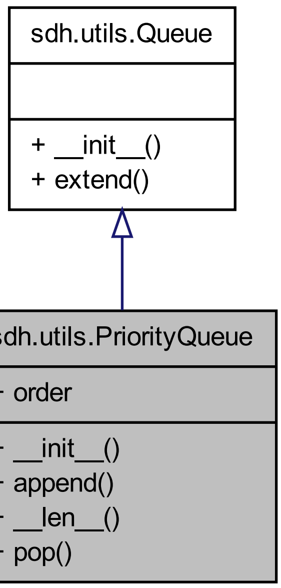
The documentation for this class was generated from the following file:

- sdh/utils.py

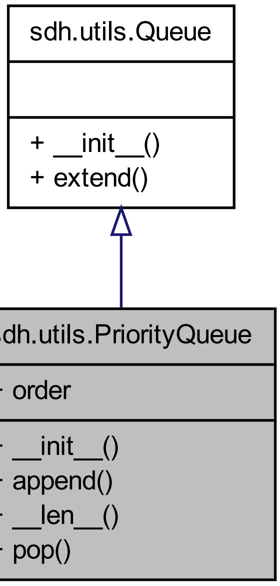
10.35 sdh.utils.PriorityQueue Class Reference

A queue in which the minimum (or maximum) element (as determined by f and order) is returned first.

Inheritance diagram for sdh.utils.PriorityQueue:



Collaboration diagram for sdh.utils.PriorityQueue:



Public Member Functions

- def __init__
- def append
- def __len__
- def pop

Public Attributes

- order

10.35.1 Detailed Description

A queue in which the minimum (or maximum) element (as determined by f and order) is returned first. If order is min, the item with minimum f(x) is returned first; if order is max, then it is the item with maximum f(x).

10.35.2 Constructor & Destructor Documentation

10.35.2.1 def sdh.utils.PriorityQueue.__init__ (*self*, *order* = `min`, *f* = `lambda x: x`)

10.35.3 Member Function Documentation

10.35.3.1 def sdh.utils.PriorityQueue.__len__ (*self*)

10.35.3.2 def sdh.utils.PriorityQueue.append (*self*, *item*)

10.35.3.3 def sdh.utils.PriorityQueue.pop (*self*)

10.35.4 Member Data Documentation

10.35.4.1 sdh.utils.PriorityQueue.order

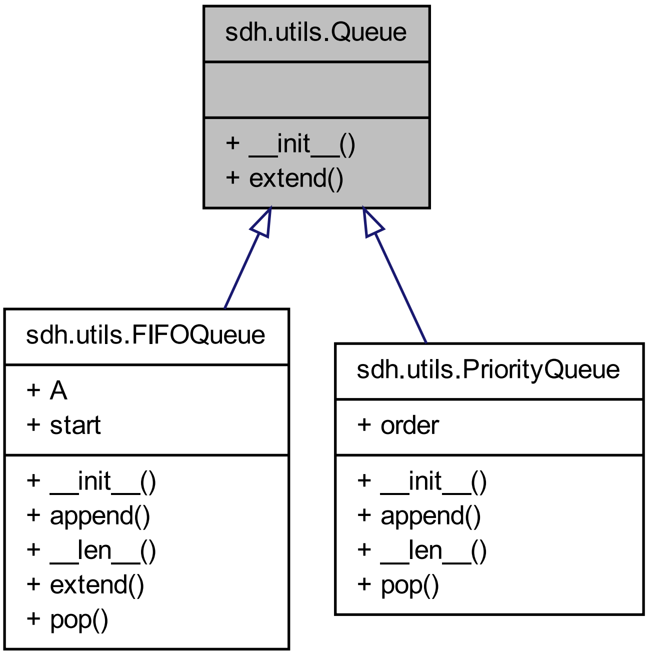
The documentation for this class was generated from the following file:

- sdh/utils.py

10.36 sdh.utils.Queue Class Reference

Queue is an abstract class/interface.

Inheritance diagram for sdh.utils.Queue:



Public Member Functions

- def __init__
- def extend

10.36.1 Detailed Description

Queue is an abstract class/interface. There are three types: Stack(): A Last In First Out Queue. FIFOQueue(): A First In First Out Queue. PriorityQueue(lt): Queue where items are sorted by lt, (default <). Each type supports the following methods and functions: q.append(item) -- add an item to the queue q.extend(items) -- equivalent to: for item in items: q.append(item) q.pop() -- return the top item from the queue len(q) -- number of items in q (also q.__len()) Note that isinstance(Stack(), Queue) is false, because we implement stacks as lists. If Python ever gets interfaces, Queue will be an interface.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 def sdh.utils.Queue.__init__ (*self*)

Reimplemented in sdh.utils.FIFOQueue.

10.36.3 Member Function Documentation

10.36.3.1 def sdh.utils.Queue.extend (*self, items*)

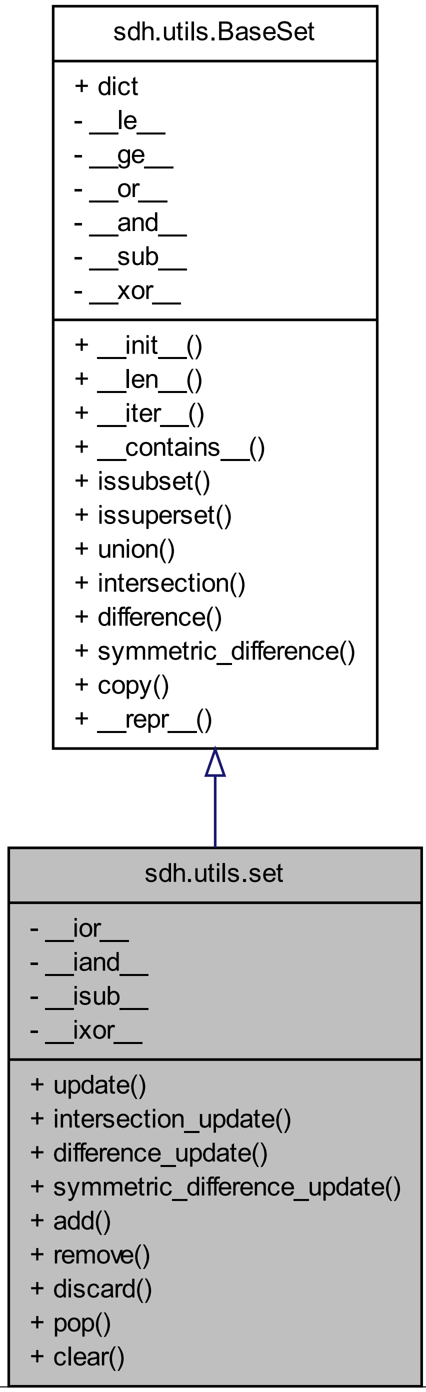
Reimplemented in sdh.utils.FIFOQueue.

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

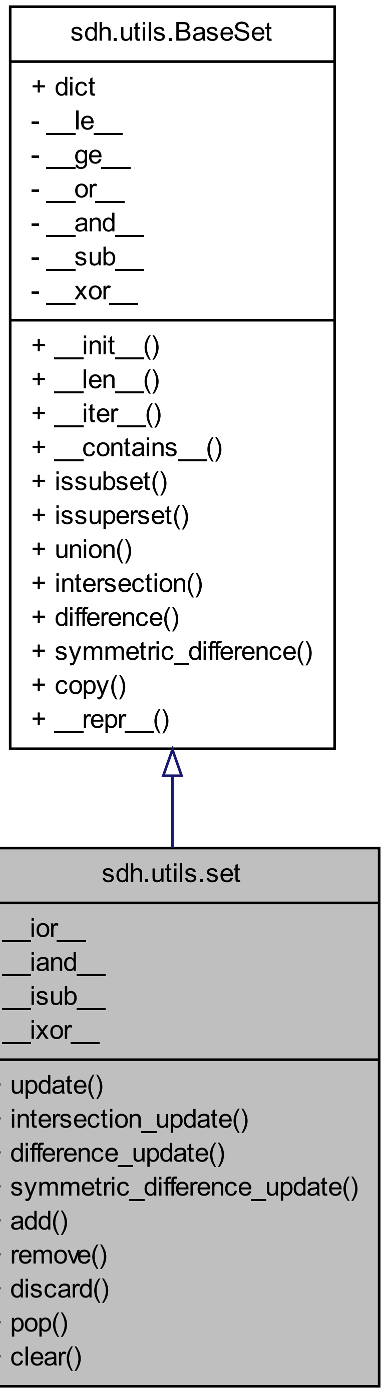
- sdh/utils.py

10.37 sdh.utils.set Class Reference

Inheritance diagram for sdh.utils.set:



Collaboration diagram for sdh.utils.set:



Public Member Functions

- def update
- def intersection_update
- def difference_update
- def symmetric_difference_update
- def add
- def remove
- def discard
- def pop
- def clear

10.37.1 Member Function Documentation

10.37.1.1 def sdh.utils.set.add (*self*, *element*)

10.37.1.2 def sdh.utils.set.clear (*self*)

10.37.1.3 def sdh.utils.set.difference_update (*self*, *other*)

10.37.1.4 def sdh.utils.set.discard (*self*, *element*)

10.37.1.5 def sdh.utils.set.intersection_update (*self*, *other*)

10.37.1.6 def sdh.utils.set.pop (*self*)

10.37.1.7 def sdh.utils.set.remove (*self*, *element*)

10.37.1.8 def sdh.utils.set.symmetric_difference_update (*self*, *other*)

10.37.1.9 def sdh.utils.set.update (*self*, *other*)

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

- sdh/utils.py

10.38 sdh.utils.Struct Class Reference

Create an instance with argument=value slots.

Public Member Functions

- def __init__
- def __cmp__
- def __repr__

10.38.1 Detailed Description

Create an instance with argument=value slots. This is for making a lightweight object whose class doesn't matter.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 def sdh.utils.Struct.__init__ (*self*, *entries*)

10.38.3 Member Function Documentation

10.38.3.1 def sdh.utils.Struct.__cmp__ (*self*, *other*)

10.38.3.2 def sdh.utils.Struct.__repr__ (*self*)

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

- sdh/utils.py

10.39 sdh.canserial.tCANSerial Class Reference

Simple wrapper class to access an ESD CAN port like a serial port as a file like object.

Public Member Functions

- def __init__

  Create a tCANSerial object for communicating via ESD card on CAN bus.

- def GetTimeout

  helper function to set property timeout

- def SetTimeout

  helper function to get property timeout

- def read

  read length bytes from CAN and return them as as string.

- def write

  Write string s to CAN.

- def flush

  This is a no-op for now.

- def close

close the CAN communication object

- def readline

  read a complete line (terminated by the eol character sequence) from CAN and return it as string.

Public Attributes

- RxTO

Properties

- timeout = property(GetTimeout, SetTimeout, None, "The timeout for reading in seconds. None == Wait for ever, 0.0 == return immediately.")

10.39.1 Detailed Description

Simple wrapper class to access an ESD CAN port like a serial port as a file like object.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 def sdh.canserial.tCANSerial.__init__ (*self*, *id_read*, *id_write*, *baudrate*, *net*, *timeout* = `2.0`)

Create a tCANSerial object for communicating via ESD card on CAN bus.

Parameters

self	- the instance of the class that this function operates on (the "object")
id_read	will be used as the CAN ID to listen to,
id_write	will be used as the CAN ID to write to,
baudrate	is in bit/s
net	is the ESD CAN net number
timeout	- timeout in seconds timeout==None => make read() / readline() wait for ever timeout==0.0 => make read() / readline() return immediately with whatever is available timeout==else => use the given timeout for read() and readline()

10.39.3 Member Function Documentation

10.39.3.1 def sdh.canserial.tCANSerial.close (*self*)

close the CAN communication object

10.39.3.2 def sdh.canserial.tCANSerial.flush (*self*)

This is a no-op for now.

Just for compatibility with the file like interface.

10.39.3.3 def sdh.canserial.tCANSerial.GetTimeout (*self*)

helper function to set property timeout

10.39.3.4 def sdh.canserial.tCANSerial.read (*self, length*)

read *length* bytes from CAN and return them as as string.

The waiting time for that many bytes depends on the setting of self.timeout and self._-return_on_less

10.39.3.5 def sdh.canserial.tCANSerial.readline (*self, eol* = '\n')

read a complete line (terminated by the *eol* character sequence) from CAN and return it as string.

10.39.3.6 def sdh.canserial.tCANSerial.SetTimeout (*self, value*)

helper function to get property timeout

10.39.3.7 def sdh.canserial.tCANSerial.write (*self, s*)

Write string *s* to CAN.

10.39.4 Member Data Documentation

10.39.4.1 sdh.canserial.tCANSerial.RxTO

10.39.5 Property Documentation

10.39.5.1 sdh.canserial.tCANSerial.timeout = property(GetTimeout, SetTimeout, None, "The timeout for reading in seconds. None == Wait for ever, 0.0 == return immediately.") [static]

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

- sdh/canserial.py

10.40 sdh.dbg.tDBG Class Reference

A class to print debug messages.

Public Member Functions

- def __init__
- def SetFlag

    Set debug_flag of this tDBG object to flag.

- def GetFlag

    Get debug_flag of this tDBG object.

- def SetColor

    Set debug_color of this tDBG object to color.

- def SetOutput

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

- def GetOutput

    Get output of this tDBG object, which is a file like object like sys.stderr.

- def SetAddNewline

    Set the do_add_newline flag of this tDBG object to flag.

- def PDM

    Print debug messages "msgs" in the color set with SetColor, but only if self.debug_flag is True.

- def __lshift__

    C++ stream like printing: d = tDBG(True) d << "bla" << "blu %s %d" % (bli,42) << True << 0815.

- def __repr__
- def var

    Print name and value of variables named in args.

- def flush

    flush output stream

Public Attributes

- debug_flag
- debug_color
- output
- do_add_newline

10.40.1 Detailed Description

A class to print 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

Example: import dbg d = dbg.tDBG(True) g = dbg.tDBG(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"

10.40.2 Constructor & Destructor Documentation

10.40.2.1 def sdh.dbg.tDBG.__init__ (*self*, *flag* = `False`, *color* = `'red'`, *fd* = `sys.stderr`)

10.40.3 Member Function Documentation

10.40.3.1 def sdh.dbg.tDBG.__lshift__ (*self*, *msg*)

C++ stream like printing: d = tDBG(True) d << "bla" << "blu %s %d" % (bli,42) << True << 0815.

10.40.3.2 def sdh.dbg.tDBG.__repr__ (*self*)

10.40.3.3 def sdh.dbg.tDBG.flush (*self*)

flush output stream

10.40.3.4 def sdh.dbg.tDBG.GetFlag (*self*)

Get debug_flag of this tDBG object.

10.40.3.5 def sdh.dbg.tDBG.GetOutput (*self*)

Get output of this tDBG object, which is a file like object like sys.stderr.

10.40.3.6 def sdh.dbg.tDBG.PDM (*self*, *msgs*)

Print debug messages "msgs" in the color set with SetColor, but only if self.debug_flag is True.

10.40.3.7 def sdh.dbg.tDBG.SetAddNewline (*self*, *flag*)

Set the do_add_newline flag of this tDBG object to flag.

If True then a newline is automatically printed after each printed debug message (like in std python print), else not (like in C/C++).

10.40.3.8 def sdh.dbg.tDBG.SetColor (*self*, *color*)

Set debug_color of this tDBG object to color.

color is a string like "red", see util.py for valid names.

10.40.3.9 def sdh.dbg.tDBG.SetFlag (*self*, *flag*)

Set debug_flag of this tDBG object to flag.

After setting the flag to True debug messages are printed, else not.

10.40.3.10 def sdh.dbg.tDBG.SetOutput (*self*, *fd*)

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

10.40.3.11 def sdh.dbg.tDBG.var (*self*, *args*)

Print name and value of variables named in args.

This will print NAME = VALUE pairs for all variables NAME in args. args is a list of strings where each string is the name of a variable in the context of the caller or args is a string with space separated names of variables in the context of the caller.

d = tDBG(True) v = 42 s = "test" d.var("v", "s") d.var("v s")

Both lines will print "v = 42, s = test"

10.40.4 Member Data Documentation

10.40.4.1 sdh.dbg.tDBG.debug_color

10.40.4.2 sdh.dbg.tDBG.debug_flag

10.40.4.3 sdh.dbg.tDBG.do_add_newline

10.40.4.4 sdh.dbg.tDBG.output

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

- sdh/dbg.py

10.41 sdh.util.tMyOptionParser Class Reference

OptionParser with some default options already set: -d | --debug turn on debug (set options.debug flag) -v | --version print version and exit.

Inherits OptionParser.

Public Member Functions

- def ShowVersion
- def __init__

  Create a tMyOptParser instance.

Public Attributes

- version

10.41.1 Detailed Description

OptionParser with some default options already set: -d | --debug turn on debug (set options.debug flag) -v | --version print version and exit.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 def sdh.util.tMyOptionParser.__init__ (*self*, *usage* = " ", *version* = " ")

Create a tMyOptParser instance.

usage has the usual meaning and version is the string that is printed when -v | --version option is set

10.41.3 Member Function Documentation

10.41.3.1 def sdh.util.tMyOptionParser.ShowVersion (*self*, *option*, *opt*, *value*, *parser*)

10.41.4 Member Data Documentation

10.41.4.1 sdh.util.tMyOptionParser.version

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

- sdh/util.py

10.42 sdh.tcpserial.tTCPSerial Class Reference

Simple wrapper class to access a TCP port like a serial port as a file like object.

Public Member Functions

- def __init__

  Create a tTCPSerial object for communicating via TCP/IP.

- def GetTimeout

  helper function to set property timeout

- def SetTimeout

  helper function to get property timeout

- def read

  read length bytes from the TCP socket and return them as as string.

- def write

  Write string s to TCP socket.

- def flush

  This is a no-op for now.

- def close

  close the CAN communication object

- def readline

  read a complete line (terminated by the eol character sequence) from CAN and return it as string.

Properties

- timeout = property(GetTimeout, SetTimeout, None, "The timeout for reading in seconds. None == Wait for ever, 0.0 == return immediately.")

10.42.1 Detailed Description

Simple wrapper class to access a TCP port like a serial port as a file like object.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 def sdh.tcpserial.tTCPSerial.__init__ (*self*, *tcp_adr* = "192.168.1.1", *tcp_port* = 23, *timeout* = 2.0)

Create a tTCPSerial object for communicating via TCP/IP.

Parameters

self	- the instance of the class that this function operates on (the "object")
tcp_adr	- the TCP adress of the SDH as IPv4 numeric address or hostname
tcp_port	- the TCP port number of the SDH,
timeout	- timeout in seconds timeout==None => make read() / readline() wait for ever timeout==0.0 => make read() / readline() return immediately with whatever is available timeout==else => use the given timeout for read() and readline()

10.42.3 Member Function Documentation

10.42.3.1 def sdh.tcpserial.tTCPSerial.close (*self*)

close the CAN communication object

10.42.3.2 def sdh.tcpserial.tTCPSerial.flush (*self*)

This is a no-op for now.

Just for compatibility with the file like interface.

10.42.3.3 def sdh.tcpserial.tTCPSerial.GetTimeout (*self*)

helper function to set property timeout

10.42.3.4 def sdh.tcpserial.tTCPSerial.read (*self*, *length*)

read *length* bytes from the TCP socket and return them as as string.

The waiting time for that many bytes depends on the setting of timeout

10.42.3.5 def sdh.tcpserial.tTCPSerial.readline (*self*, *eol* = '\n')

read a complete line (terminated by the *eol* character sequence) from CAN and return it as string.

10.42.3.6 def sdh.tcpserial.tTCPSerial.SetTimeout (*self*, *value*)

helper function to get property timeout

10.42.3.7 def sdh.tcpserial.tTCPSerial.write (*self*, *s*)

Write string *s* to TCP socket.

10.42.4 Property Documentation

10.42.4.1 sdh.tcpserial.tTCPSerial.timeout = property(GetTimeout, SetTimeout, None, "The timeout for reading in seconds. None == Wait for ever, 0.0 == return immediately.") [static]

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

- sdh/tcpserial.py

Chapter 11

File Documentation

11.1 demo/demo-benchmark.py File Reference

Simple script to do grasping using tactile sensor info feedback. See demo-benchmark.__doc__ and the online help ("-h" or "--help") for a list of available options.

Packages

- package demo-benchmark

Python specific variables

Some definitions that describe the script for python

- string demo-benchmark.__doc__

  The docstring describing the purpose of the script:

- string demo-benchmark.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-benchmark.__url__ = "http://www.schunk.com"
- string demo-benchmark.__version__ = "$Id: demo-benchmark.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-benchmark.__copyright__ = "Copyright (c) 2011 SCHUNK GmbH & Co. KG"
- int demo-benchmark.DEMO_BENCHMARK_USE_COMBINED_SET_GET = 1
- def demo-benchmark.CreateOptionParser

  Command line option handling:

- def demo-benchmark.GotoPose
- def demo-benchmark.Flat

  print flat representation of iterable l ([1,2,3] yields "1, 2, 3")

- def demo-benchmark.main

  The main function.

11.1.1 Detailed Description

Simple script to do grasping using tactile sensor info feedback. See demo-benchmark._-_doc__ and the online help ("-h" or "--help") for a list of available options.

11.1.2 General file information

Author

Dirk Osswald

Date

2011-02-08

11.1.3 Copyright

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

11.2 demo/demo-calc-workspace.py File Reference

Output a data file with xyz fingertip positions for all possible angles.

Packages

- package demo-calc-workspace

Functions

- def demo-calc-workspace.Print

Variables

- tuple demo-calc-workspace.parser

  Command line option handling:

- tuple demo-calc-workspace.types = dict(all=0, contour=1)
- string demo-calc-workspace.dest = "step0"

- string demo-calc-workspace.help = "Set step width for finger axis angle 0 to STEP, default=5."
- tuple demo-calc-workspace.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple demo-calc-workspace.hand = sdh.cSDH(options=options.__dict__)

  The actual script code:

- float demo-calc-workspace.phi = 90.0

Python specific variables

Some definitions that describe the script for python

Output a data file with xyz fingertip positions for all possible angles

- string demo-calc-workspace.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-calc-workspace.__url__ = "http://www.schunk.com"
- string demo-calc-workspace.__version__ = "$Id: demo-calc-workspace.py 4355 2009-05-04 17:17:39Z Osswald2 $"
- string demo-calc-workspace.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.2.1 Detailed Description

Output a data file with xyz fingertip positions for all possible angles.

11.2.2 General file information

Author

Dirk Osswald

Date

2007-02-06

Will not connect to a real SDH.

Usefulle e.g. as input for gnuplot's splot command:

- In a shell type:

```
demo-calc-workspace.py > workspace.dat
```

- Then in a gnuplot command line:

```
splot 'workspace.dat' using 4:5:6, 'workspace.dat' using 7:8:9, 'workspace.da
  t' using 10:11:12
```

Start the script with "`-h`" or "`--help`" command line option to see the online help.

11.2.3 Copyright

11.3 demo/demo-contact-grasping.py File Reference

Simple script to do grasping using tactile sensor info feedback. See demo-contact-grasping.__doc__ and the online help ("-h" or "--help") for a list of available options.

Packages

- package demo-contact-grasping

Python specific variables

Some definitions that describe the script for python

- string demo-contact-grasping.__doc__

  The docstring describing the purpose of the script:

- string demo-contact-grasping.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-contact-grasping.__url__ = "http://www.schunk.com"
- string demo-contact-grasping.__version__ = "$Id: demo-contact-grasping.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-contact-grasping.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def demo-contact-grasping.CreateOptionParser

  Command line option handling:

- def demo-contact-grasping.GotoStartPose
- def demo-contact-grasping.main

  The main function.

11.3.1 Detailed Description

Simple script to do grasping using tactile sensor info feedback. See demo-contact-grasping.__doc__ and the online help ("-h" or "--help") for a list of available options.

11.3.2 General file information

Author

Dirk Osswald

Date

2007-05-08

11.3.3 Copyright

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

11.4 demo/demo-dsa.py File Reference

Simple script to access tactile sensors of SDH. See demo-dsa.__doc__ and online help ("-h" or "--help") for available options.

Classes

- class demo-dsa.cMovingAverage

  Some additional classes and functions.

Packages

- package demo-dsa

Python specific variables

Some definitions that describe the script for python

- string demo-dsa.__doc__

  The docstring describing the purpose of the script:

- string demo-dsa.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-dsa.__url__ = "http://www.schunk.com"
- string demo-dsa.__version__ = "$Id: demo-dsa.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-dsa.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def demo-dsa.CreateOptionParser

  Create an option parser specifically for this demo program.

- def demo-dsa.main

  The main function.

11.4.1 Detailed Description

Simple script to access tactile sensors of SDH. See demo-dsa.__doc__ and online help ("-h" or "--help") for available options.

11.4.2 General file information

Author

Dirk Osswald

Date

2007-05-08

11.4.3 Copyright

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

11.5 demo/demo-GetAxisActualAngle.py File Reference

Print current actual axis angles from SDH. See demo-GetAxisActualAngle.__doc__ and online help ("-h" or "--help") for available options.

Packages

- package demo-GetAxisActualAngle

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-GetAxisActualAngle.__doc__
  The docstring describing the purpose of the script:
- string demo-GetAxisActualAngle.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-GetAxisActualAngle.__url__ = "http://www.schunk.com"
- string demo-GetAxisActualAngle.__version__ = "$Id: demo-GetAxisActualAngle.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-GetAxisActualAngle.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-GetAxisActualAngle.parser
  Command line option handling:
- string demo-GetAxisActualAngle.dest = "period"

- string demo-GetAxisActualAngle.help = "Time period of measurements in seconds. The default of '0' means: report once only."
- tuple demo-GetAxisActualAngle.dbg = sdh.dbg.tDBG(flag=options.debug_-level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple demo-GetAxisActualAngle.hand = sdh.cSDH(options=options.__dict_-_)

  The actual script code:

- tuple demo-GetAxisActualAngle.t = hand.MoveHand(sequ=False)
- tuple demo-GetAxisActualAngle.start = sdh.time.time()
- tuple demo-GetAxisActualAngle.a_angles = hand.GetAxisActualAngle(sdh.All)
- tuple demo-GetAxisActualAngle.a_velocities = hand.GetAxisActualVelocity(sdh.All)
- tuple demo-GetAxisActualAngle.now = sdh.time.time()
- tuple demo-GetAxisActualAngle.xyz = hand.GetFingerXYZ(fi, None)

11.5.1 Detailed Description

Print current actual axis angles from SDH. See demo-GetAxisActualAngle.__doc__ and online help ("-h" or "--help") for available options.

11.5.2 General file information

Author

Dirk Osswald

Date

2007-01-29

Start the script with "`-h`" or "`--help`" command line option to see the online help.

11.5.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.6 demo/demo-gui.py File Reference

Simple GUI (Graphical User Interface) to control an SDH. See demo-gui.__doc__ and online help ("-h" or "--help") for available options.

Classes

- class demo-gui.cTkSDHFinger

  A widget for a single finger.

- class demo-gui.cTkSDHSavePose

  A widget to save restore a single pose.

- class demo-gui.cTkSDHSavePoses

  A widget to save/restore all poses.

- class demo-gui.cTkSDHGrip

  A widget to access the grip skills stored in the SDH.

- class demo-gui.cTkSDHButtonBox

  A simple box for buttons.

- class demo-gui.cTkSDHMenu

  The Menu for the application.

- class demo-gui.cTkSDHPID

  Toplevel window to show and adjust the pid parameters of an SDH.

- class demo-gui.cTkSDHCurrent

  Toplevel window to show and adjust the motor current parameters of an SDH.

- class demo-gui.cTkSDHApplication

  The "Application" class of the simple SDH GUI.

- class demo-gui.cTkSDHInterfaceSelectorToplevel

  A toplevel widget class, used to select the communication interface to the SDH.

Packages

- package demo-gui

Python specific variables

Some definitions that describe the script for python

- string demo-gui.__doc__

  The docstring describing the purpose of the script:

- string demo-gui.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"

- string demo-gui.__url__ = "http://www.schunk.com"
- string demo-gui.__version__ = "$Id: demo-gui.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string demo-gui.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- demo-gui.we_have_can = True
- demo-gui.hand = None

  global variables

- demo-gui.dbg = None
- demo-gui.options = None
- demo-gui.root = None
- tuple demo-gui.persistent_settings = sdh.util.GetPersistantDict(name=".demo-gui-startsettings", cdbg = dbg)
- string demo-gui.schunk_logo
- def demo-gui.main

11.6.1 Detailed Description

Simple GUI (Graphical User Interface) to control an SDH. See demo-gui.__doc__ and online help ("-h" or "--help") for available options.

11.6.2 General file information

Author

Dirk Osswald

Date

2007-01-30

11.6.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.7 demo/demo-radians.py File Reference

Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control).

Packages

- package demo-radians

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-radians.__doc__

  The docstring describing the purpose of the script:

- string demo-radians.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-radians.__url__ = "http://www.schunk.com"
- string demo-radians.__version__ = "$Id: demo-radians.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string demo-radians.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-radians.parser

  Command line option handling:

- tuple demo-radians.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple demo-radians.hand = sdh.cSDH(options=options.__dict__)
- tuple demo-radians.faa = hand.GetFingerActualAngle(0)

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple demo-radians.fta = list(faa)

11.7.1 Detailed Description

Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control). See demo-radians.__doc__ and online help ("-h" or "--help") for available options.

This script contains only the very basicst use of sdh.py features. For more sophisticated applications see the other demo-∗.py scripts, or of course the html/pdf documentation.

11.8 demo/demo-simple.py File Reference

Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control).

Packages

- package demo-simple

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-simple.__doc__

  The docstring describing the purpose of the script:

- string demo-simple.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-simple.__url__ = "http://www.schunk.com"
- string demo-simple.__version__ = "$Id: demo-simple.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string demo-simple.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-simple.parser

  Command line option handling:

- tuple demo-simple.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple demo-simple.hand = sdh.cSDH(options=options.__dict__)
- tuple demo-simple.faa = hand.GetFingerActualAngle(0)

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple demo-simple.fta = list(faa)

11.8.1 Detailed Description

Very simple demonstration of the sdh python package: Make the SDH move one finger with "pose" controller type (coordinated position control). See demo-simple.__doc__ and online help ("-h" or "--help") for available options.

This script contains only the very basicst use of sdh.py features. For more sophisticated applications see the other demo-∗.py scripts, or of course the html/pdf documentation.

11.9 demo/demo-simple2.py File Reference

Very simple demonstration of the sdh python package.

Packages

- package demo-simple2

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-simple2.__doc__

  The docstring describing the purpose of the script:

- string demo-simple2.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-simple2.__url__ = "http://www.schunk.com"
- string demo-simple2.__version__ = "$Id: demo-simple2.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-simple2.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-simple2.parser

  Command line option handling:

- tuple demo-simple2.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple demo-simple2.hand = sdh.cSDH(options=options.__dict__)
- int demo-simple2.iFinger = 0

  do some prepartions: Switch to "pose" controller mode and set default velocities first:

- tuple demo-simple2.faa = hand.GetFingerActualAngle(iFinger)
- tuple demo-simple2.fta = list(faa)
- tuple demo-simple2.t = hand.MoveFinger(iFinger, False)

11.9.1 Detailed Description

Very simple demonstration of the sdh python package. Make the SDH move and stop one finger. See demo-simple2.__doc__ and online help ("-h" or "--help") for available options.

This script contains only the very basicst use of sdh.py features. For more sophisticated applications see the other demo-∗.py scripts, or of course the html/pdf documentation.

11.10 demo/demo-simple3.py File Reference

Very simple demonstration of the sdh python package: Make the SDH move 3 axes.

Packages

- package demo-simple3

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-simple3.__doc__

    The docstring describing the purpose of the script:

- string demo-simple3.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-simple3.__url__ = "http://www.schunk.com"
- string demo-simple3.__version__ = "$Id: demo-simple3.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-simple3.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-simple3.parser

    Command line option handling:

- tuple demo-simple3.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

    An object to print script-level debug messages, if requested.

- tuple demo-simple3.hand = sdh.cSDH(options=options.__dict__)

11.10.1 Detailed Description

Very simple demonstration of the sdh python package: Make the SDH move 3 axes. See demo-simple3.__doc__ and online help ("-h" or "--help") for available options.

This script contains only the very basicst use of sdh.py features. For more sophisticated applications see the other demo-∗.py scripts, or of course the html/pdf documentation.

11.11 demo/demo-tactile.py File Reference

Simple GUI to visualize tactile sensors of SDH. See demo-tactile.__doc__ and the online help ("-h" or "--help") for available options.

Classes

- class demo-tactile.cTkSDHTactileApplication

    The "Application" class of demo-tactile.py, the simple SDH tactile visualizer.

Packages

- package demo-tactile

Python specific variables

Some definitions that describe the script for python

- string demo-tactile.__doc__

  The docstring describing the purpose of the script:

- string demo-tactile.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-tactile.__url__ = "http://www.schunk.com"
- string demo-tactile.__version__ = "$Id: demo-tactile.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-tactile.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-tactile._dbg = sdh.dbg.tDBG(True)
- def demo-tactile.main

11.11.1 Detailed Description

Simple GUI to visualize tactile sensors of SDH. See demo-tactile.__doc__ and the online help ("-h" or "--help") for available options.

11.11.2 General file information

Author

Dirk Osswald

Date

2007-03-12

11.11.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.12 demo/demo-temperature.py File Reference

Print measured temperatures of SDH. See demo-temperature.__doc__ and online help ("-h" or "--help") for available options.

Packages

- package demo-temperature

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-temperature.__doc__

  The docstring describing the purpose of the script:

- string demo-temperature.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-temperature.__url__ = "http://www.schunk.com"
- string demo-temperature.__version__ = "$Id: demo-temperature.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-temperature.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-temperature.parser

  Command line option handling:

- string demo-temperature.dest = "period"
- string demo-temperature.help = "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 beginning of every line"
- tuple demo-temperature.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_output)

  An object to print script-level debug messages, if requested.

- tuple demo-temperature.hand = sdh.cSDH(options=options.__dict__)

  The actual script code:

- tuple demo-temperature.start = sdh.time.time()
- tuple demo-temperature.L = hand.GetTemperature()

11.12.1 Detailed Description

Print measured temperatures of SDH. See demo-temperature.__doc__ and online help ("-h" or "--help") for available options.

11.12.2 General file information

Author

Dirk Osswald

Date

2007-01-18

11.12.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.13 demo/demo-velocity-acceleration.py File Reference

Demonstration script of the sdh python package: Make the SDH move one finger in "velocity with acceleration ramp" control mode.

Packages

- package demo-velocity-acceleration

Variables

Python specific variables

Some definitions that describe the script for python

- string demo-velocity-acceleration.__doc__
  The docstring describing the purpose of the script:
- string demo-velocity-acceleration.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-velocity-acceleration.__url__ = "http://www.schunk.com"
- string demo-velocity-acceleration.__version__ = "$Id: demo-velocity-acceleration.py 10351 2013-06-18 16:28:14Z Osswald2 $"
- string demo-velocity-acceleration.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-velocity-acceleration.parser
  Command line option handling:
- tuple demo-velocity-acceleration.dbg = sdh.dbg.tDBG(flag=options.debug_-level>0, fd=options.debug_output)
  An object to print script-level debug messages, if requested.
- tuple demo-velocity-acceleration.hand = sdh.cSDH(options=options.__dict_-_)
- int demo-velocity-acceleration.axis_index = 2
  Preparations: Move the hand to a pose that is adequate for this demo:
- int demo-velocity-acceleration.velocity = 40
- demo-velocity-acceleration.position_reached = False

11.13.1 Detailed Description

Demonstration script of the sdh python package: Make the SDH move one finger in "velocity with acceleration ramp" control mode. See demo-simple.__doc__ and online help ("-h" or "--help") for available options.

11.14 demo/demo-workspace.py File Reference

Move fingers to show workspace of SDH. (Python demo script using the sdh.py import library.)

Packages

- package demo-workspace

Variables

Python specific variables

Some definitions that describe the script for python

Move fingers to show workspace of SDH. (Python demo script using the sdh.py import library.)

- string demo-workspace.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string demo-workspace.__url__ = "http://www.schunk.com"
- string demo-workspace.__version__ = "$Id: demo-workspace.py 6269 2010-12-03 11:46:13Z Osswald2 $"
- string demo-workspace.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple demo-workspace.parser

  Command line option handling:

- tuple demo-workspace.dbg = sdh.dbg.tDBG(flag=options.debug_level>0, fd=options.debug_-output)

  An object to print script-level debug messages, if requested.

- tuple demo-workspace.hand = sdh.cSDH(options=options.__dict__)

  The actual script code.

- tuple demo-workspace.ata = map(hand.uc_angle.ToExternal, ata)
- tuple demo-workspace.t = hand.MoveHand()

11.14.1 Detailed Description

Move fingers to show workspace of SDH. (Python demo script using the sdh.py import library.)

11.14.2 General file information

Author

Dirk Osswald

Date

2007-01-29

Start the script with "`-h`" or "`--help`" command line option to see the online help.

11.14.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.15 demo/miniterm.py File Reference

Very simple serial terminal.

Classes

- class miniterm.cTkSDHInterfaceSelectorFrame

  A toplevel widget class, used to interactively select the communication interface of the miniterm.py app on start.

Packages

- package miniterm

Functions

- def miniterm.GetColor

  return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_-back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

- def miniterm.GetPrompt
- def miniterm.hex2

  Return the hexadecimal representation of an integer or long integer with 2 digits (hex2(5)->0x05)

- def miniterm.cls

  Clear screen.

- def miniterm.reader

  loop forever and copy serial->console

- def miniterm.StringToInt

  return int of string s, e.g.

- def miniterm.HexStringToInt
- def miniterm.SendFromFile

  send data from file filename to serial port.

- def miniterm.writer

    loop and copy console->serial until EOF character is found

- def miniterm.usage
- def miniterm.Exit

    if wait_for_key is False then just call exit.

- def miniterm.main

Variables

- list miniterm.d = os.environ["HOME"]
- tuple miniterm.histfile = os.path.join(d, ".minitermhist")
- tuple miniterm.prefix_keyboard = GetColor("blue")
- tuple miniterm.suffix_keyboard = GetColor("normal")
- tuple miniterm.prefix_serialin = GetColor("normal")
- tuple miniterm.suffix_serialin = GetColor("normal")
- tuple miniterm.prefix_message = GetColor("green")
- tuple miniterm.suffix_message = GetColor("normal")
- tuple miniterm.prefix_error = GetColor("red")
- tuple miniterm.suffix_error = GetColor("normal")
- tuple miniterm.prefix_warning = GetColor("magenta")
- tuple miniterm.suffix_warning = GetColor("normal")
- string miniterm.VT100_CLR_SCREEN = "\x1b[2J"
- string miniterm.EXITCHARACTER = '\x04'
- int miniterm.CONVERT_CRLF = 2
- int miniterm.CONVERT_CR = 1
- int miniterm.CONVERT_LF = 0
- int miniterm.eModeAscii = 0
- int miniterm.eModeNumeric = 1
- int miniterm.eModeHexNumeric = 2
- miniterm.mode = eModeAscii
- miniterm.additional_ascii = False
- int miniterm.numeric_length = 8
- miniterm.prompt = None
- miniterm.input_log_file = None
- miniterm.inputfilename = None
- miniterm.g_exiting = False
- miniterm.g_reader_thread = None
- miniterm.serialport = None
- string miniterm.online_help
- miniterm.convert_outgoing = CONVERT_CRLF

11.15.1 Detailed Description

Very simple serial terminal.

11.15.2 General file information

Author

Dirk Osswald

Date

2007-01-29

Source: pyserial the system independent serial port access module An input line is read with readline and sent to the serial port on return. Commands can be edited with the cursor, delete, backspace, insert keys. Previous commands, even from a previous session can be reached with cursor up or CTRL-R. A history of lines is saved in ~/.minitermhist and reread on the next invocation.

Input characters are sent directly (only LF -> CR/LF/CRLF translation is done, if desired), received characters are displayed as is (or as trough pythons repr, useful for debug purposes) Baudrate and echo configuartion is done through globals

As communication channels the following are available:

- "normal" RS232 ports
- jtag_uart via the nios2-terminal program (if the jtagserial module is available).
- CAN where data is sent on one ID and received on another (if the canserial module is available and an ESD CAN card, native Windows only)

Start the script with `"-h"` or `"--help"` command line option to see the online help.

11.15.3 Copyright

11.16 demo/sdh-ping.py File Reference

Measure response time of SDH See sdh-ping.__doc__ and online help ("-h" or "--help") for available options.

Packages

- package sdh-ping

Python specific variables

Some definitions that describe the script for python

- string sdh-ping.__doc__

  The docstring describing the purpose of the script:

- string sdh-ping.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh-ping.__url__ = "http://www.schunk.com"
- string sdh-ping.__version__ = "$Id: sdh-ping.py 6270 2010-12-03 11:49:03Z Osswald2 $"
- string sdh-ping.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- def sdh-ping.GetMedian
- def sdh-ping.T2MS
- def sdh-ping.avg
- def sdh-ping.main

11.16.1 Detailed Description

Measure response time of SDH See sdh-ping.__doc__ and online help ("-h" or "--help") for available options.

11.17 doc/onlinehelp-demo-benchmark.dox File Reference

11.17.1 Detailed Description

11.18 doc/onlinehelp-demo-calc-workspace.dox File Reference

11.18.1 Detailed Description

11.19 doc/onlinehelp-demo-contact-grasping.dox File Reference

11.19.1 Detailed Description

11.20 doc/onlinehelp-demo-dsa.dox File Reference

11.20.1 Detailed Description

11.21 doc/onlinehelp-demo-GetAxisActualAngle.dox File Reference

11.21.1 Detailed Description

11.22 doc/onlinehelp-demo-gui.dox File Reference

11.22.1 Detailed Description

11.23 doc/onlinehelp-demo-radians.dox File Reference

11.23.1 Detailed Description

11.24 doc/onlinehelp-demo-simple.dox File Reference

11.24.1 Detailed Description

11.25 doc/onlinehelp-demo-simple2.dox File Reference

11.25.1 Detailed Description

11.26 doc/onlinehelp-demo-simple3.dox File Reference

11.26.1 Detailed Description

11.27 doc/onlinehelp-demo-tactile.dox File Reference

11.27.1 Detailed Description

11.28 doc/onlinehelp-demo-temperature.dox File Reference

11.28.1 Detailed Description

11.32.1 Detailed Description

Doxyfile for generating documentation for SDHLibrary python using doxygen.

11.32.2 General file information

Author

Dirk Osswald

Date

2007-06-14

11.32.3 Links

- The online documentation for Doxygen can be found at `http://www.stack.nl/~dimitri/doxyg`

11.32.4 Copyright

11.33 Makefile File Reference

Makefile for SDH SDHLibrary python project.

11.33.1 Detailed Description

Makefile for SDH SDHLibrary python project.

11.33.2 General file information

Author

Dirk Osswald

Date

2007-01-03

This makefile can install/uninstall the python package, generate 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.

11.33.3 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

11.33.4 Makefile targets

- **`all`** : generate everything
  - **`doc`** : generate all documentation (not available in distribution)
- **install** : install python package, demo scripts and documentation in a native pythonic way using distutils
- **uninstall: uninstall** previously installed stuff
- **`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
- **`test`** : run automated unit tests using py.test
- **`dos2unix`** : convert line endings from dos/windows format to unix format
- **`unix2dos`** : convert line endings from unix format to dos/windows format

11.33.5 Links

- The online documentation for `gnu make` can be found at `http://www.gnu.org/software/make/manual/m`

11.33.6 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.34 postinstall_sdh.py File Reference

Windows installer postinstall script for python distutils setup.py script. Installs shortcuts to the documentation and the actual update scripts into 'Start->Programs->SCHUNK->SDH...

Packages

- package postinstall_sdh

Functions

- def postinstall_sdh.Log
- def postinstall_sdh.Install

  define necessary functions

- def postinstall_sdh.Remove

Variables

- postinstall_sdh.do_debug = False

  simple logging mechanism for debugging the postinstall script.

- postinstall_sdh.log = None
- postinstall_sdh.args_ok = False

  "main" function that calls the functions from above according to command line

11.34.1 Detailed Description

Windows installer postinstall script for python distutils setup.py script. Installs shortcuts to the documentation and the actual update scripts into 'Start->Programs->SCHUNK->SDH...

11.34.2 General file information

Author

Dirk Osswald

Date

2008-03-25

(see `file:///D:/Programme/cygwin/usr/share/doc/python-2.5.2/html/dist/postins`
or `http://osdir.com/ml/python.ipython.user/2005-01/msg00014.html`
)

11.34.3 Copyright

- Copyright (c) 2008 SCHUNK GmbH & Co. KG

11.35 sdh/__init__.py File Reference

Initialization of the sdh package.

Packages

- package sdh

  Implementation of the python package to control a SDH (SCHUNK Dexterous Hand).

Variables

Python specific variables

Some definitions that describe the module for python

- string sdh.__doc__
- string sdh.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh.__url__ = "http://www.schunk.com"
- sdh.__version__ = release.PROJECT_RELEASE
- string sdh.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.35.1 Detailed Description

Initialization of the sdh package.

11.35.2 General file information

Author

Dirk Osswald

Date

2007-06-11

11.35.3 Copyright

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

11.36 sdh/auxiliary.py File Reference

Implementation of auxiliary variables, functions, classes.

Classes

- class sdh.auxiliary.cSDHOptionParser

  Customized OptionParser with some SDH specific options already set.

- class sdh.auxiliary.cSphere

  A class to represent sphere objects.

Packages

- package sdh.auxiliary

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::auxiliary.__doc__ = "Auxiliary variables, functions, classes for sdh package"
- string sdh::auxiliary.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::auxiliary.__url__ = "http://www.schunk.com"
- string sdh::auxiliary.__version__ = "$Id: auxiliary.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string sdh::auxiliary.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

Auxiliary variables, functions, classes

- sdh::auxiliary.has_dsa = True

  Auxiliary variables:

- float sdh::auxiliary.MIN_FLOAT = 3.40E+38
- float sdh::auxiliary.MAX_FLOAT = 3.40E+38
- def sdh::auxiliary.InIndex

  Auxiliary functions:

- def sdh::auxiliary.InRange

  Return True if v is in range [min_v .

- def sdh::auxiliary.ToRange

  Return v limited to range [min_v .

- def sdh::auxiliary.Approx

  Return True if a is approximately the same as b.

- def sdh::auxiliary.InRange_a

  Return True if in list/tuple/array v=(v1,v2,...) each v_i is in range [min_v_i..max_v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

- def sdh::auxiliary.ToRange_a

  Return list/tuple/array v=(v1,v2,...) where each v_i is limited to range [min_v_i..max_v_i] with min_v = (min_v1, min_v2,...) max_v = (max_v1, max_v2, ..)

- def sdh::auxiliary.Approx_a

  Return True if list/tuple/array a=(a1,a2,...) is approximately the same as b=(b1,b2,...).

- def sdh::auxiliary.DegToRad

  Return d in deg converted to rad.

- def sdh::auxiliary.RadToDeg

  Return r in rad converted to deg.

- def sdh::auxiliary.Square

  Return v squared (i.e.

- def sdh::auxiliary.Alltrue

  Return True if all elements of v (tuple, list, array.array).

- def sdh::auxiliary.Allmin

  Return list of min elements of v and w (tuple, list, array.array).

- def sdh::auxiliary.Allmax

  Return list of max elements of v and w (tuple, list, array.array).

- def sdh::auxiliary.AsStruct

  return dict_or_struct as struct

- def sdh::auxiliary.GetVersionInfo

  Return a string with all the version info:
  - *name and release of the calling script (PC),*
  - *name and release of the library (PC),*
  - *name of the platform (PC),*
  - *name and release of the Python executable (PC),*
  - *release and date of firmware (SDH),*
  - *release and date of SoC (SDH),*
  - *id and serial number of the SDH,*
  - *hardware and software versions and serial numbers for:*
    - *the tactile controller DSACON32m (SDH),*
    - *the tactile sensors (SDH)*

- def sdh::auxiliary.GetCommunicationInterfaceName
- def sdh::auxiliary.WriteIVFile

  Generate an OpenInventor iv file of all the objects by calling o.Toiv()

- def sdh::auxiliary.GetDevicePatterns

  Return a list of RS232 device name patterns corresponding to the current platform.

- def sdh::auxiliary.GetAvailablePorts

  Return a list of tuples (p,occupied), where p is device name of a serial port of the computer and occupied is True if the port is occupied by another application or the port is inaccessible.

- def sdh::auxiliary.GetIconPath
  Return a path to an appropriate icon for this application and OS.
- def sdh::auxiliary.PrettyStruct
  Return a string containing the name and the content of the structure s.
- def sdh::auxiliary.NumerifyRelease
  return a list of integer numbers for a release string
- def sdh::auxiliary.CompareReleases
  compare release strings rev1 and rev2.

11.36.1 Detailed Description

Implementation of auxiliary variables, functions, classes.

11.36.2 General file information

Author

Dirk Osswald

Date

2007-06-13

11.36.3 Copyright

11.37 sdh/canserial.py File Reference

Simple wrapper to access an ESD CAN port like a serial port.

Classes

- class sdh.canserial.tCANSerial
  Simple wrapper class to access an ESD CAN port like a serial port as a file like object.

Packages

- package sdh.canserial

11.37.1 Detailed Description

Simple wrapper to access an ESD CAN port like a serial port.

11.37.2 General file information

Author

Dirk Osswald

Date

2007-05-04

11.37.3 Copyright

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

11.38 sdh/dbg.py File Reference

Provides class tDBG, a class to print debug messges, see there.

Classes

- class sdh.dbg.tDBG

    A class to print debug messages.

Packages

- package sdh.dbg

11.38.1 Detailed Description

Provides class tDBG, a class to print debug messges, see there.

11.38.2 General file information

Author

Dirk Osswald

Date

2006-04-08

11.38.3 Copyright

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

11.39 sdh/dsa.py File Reference

Implementation of the python import module dsa, the interface to the DSA tactile sensor controller of an SDH.

Classes

- class sdh.dsa.cDSAError

  DSA (tactile sensor of the SDH) related error occurred.

- class sdh.dsa.cDSA

  Interface class to access the DSACON32m, the tactile sensor controller of the SDH.

Packages

- package sdh.dsa
- package dsa

  Python module to control the tactile sensors of the SDH (SCHUNK Dexterous Hand).

Functions

- def sdh::dsa.LB

  return low byte of integer value i

- def sdh::dsa.HB

  return high byte of integer value i

- def sdh::dsa.Boolify

  return True if v != 0, else False

- def sdh::dsa.CRC16

  Do cyclic redundancy check calculation.

- def sdh::dsa.UIntFromBytes

  Return an int from the bytes in list the_bytes (1,2,3,4,...,bytes) in little endian.

- def sdh::dsa.FloatFromBytes

  Return a float from the list of the_bytes.

- def sdh::dsa.FloatToBytes

  Return a list of bytes from the float the_float.

- def sdh::dsa.UInt16ToBytes

  Return a list of bytes from the UInt16 the_uint16.

Variables

- sdh::dsa.All = None
- list sdh::dsa.gCRCtbl

  The CRC table used by the DSACON32m controller.

- int sdh::dsa.CRC_INIT_VALUE = 0xffff

11.39.1 Detailed Description

Implementation of the python import module dsa, the interface to the DSA tactile sensor controller of an SDH.

11.39.2 General file information

Author

Dirk Osswald

Date

2007-05-04

11.39.3 Copyright

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

11.40 sdh/release.py File Reference

Definition and documentation of the project name and the release name ("version") of the package. The doxygen comments of the release name serve as the change log of the project.

Packages

- package sdh.release

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::release.__doc__ = """Definition and documentation of the project name and the release name ("version") for sdh package"""
- string sdh::release.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::release.__url__ = "http://www.schunk.com"
- string sdh::release.__version__ = "$Id: release.py 12281 2014-09-30 07:44:33Z Osswald2 $"
- string sdh::release.__copyright__ = "Copyright (c) 2013 SCHUNK GmbH & Co. KG"
- string sdh::release.PROJECT_NAME = "SDHLibrary-python"

  Define some variables.

- string sdh::release.FIRMWARE_RELEASE_RECOMMENDED = "0.0.3.3"
- string sdh::release.PROJECT_RELEASE = "0.0.2.9"

  Release name of the whole software project (a.k.a.

- string sdh::release.PROJECT_DATE = "2014-09-30"

  Date of the release of the software project.

11.40.1 Detailed Description

Definition and documentation of the project name and the release name ("version") of the package. The doxygen comments of the release name serve as the change log of the project.

11.40.2 General file information

Author

Dirk Osswald

Date

2007-06-13

11.40.3 Copyright

Copyright (c) 2014 SCHUNK GmbH & Co. KG

11.41 sdh/sdh.py File Reference

Implementation of the python import module sdh.

Classes

- class sdh.sdh.cSDH

  The actual SDH classes.

Packages

- package sdh.sdh

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::sdh.__doc__ = "python module with end user interface to control a SDH (SCHUNK Dexterous Hand)"
- string sdh::sdh.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::sdh.__url__ = "http://www.schunk.com"
- string sdh::sdh.__version__ = "$Id: sdh.py 11045 2013-11-27 15:12:49Z Osswald2 $"
- string sdh::sdh.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.41.1 Detailed Description

Implementation of the python import module sdh.

11.41.2 General file information

Author

Dirk Osswald

Date

2007-01-15

11.41.3 Copyright

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

11.42 sdh/sdhbase.py File Reference

Implementation of base classes to access SDH.

Classes

- class sdh.sdhbase.cSDHError
  Exception classes.
- class sdh.sdhbase.cSDHErrorCommunication
  SDH-exception: Communication error occured in the sd module.
- class sdh.sdhbase.cSDHErrorInvalidParameter
  SDH-exception: Invalid parameter(s) were given.
- class sdh.sdhbase.cSDHErrorTimeout
  SDH-exception: A (communication) timeout occured.
- class sdh.sdhbase.cSDHErrorInternalCollision
  SDH-exception: The given target angles would lead to an internal collision.
- class sdh.sdhbase.cSDHBase
  The base class to control the SCHUNK Dexterous Hand.

Packages

- package sdh.sdhbase

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::sdhbase.__doc__ = "Base classes for sdh package"
- string sdh::sdhbase.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::sdhbase.__url__ = "http://www.schunk.com"
- string sdh::sdhbase.__version__ = "$Id: sdhbase.py 6432 2011-02-08 13:53:00Z Osswald2 $"
- string sdh::sdhbase.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- sdh::sdhbase.All = None
  A constant to address all fingers/axes when used as a parameter in certain functions.

11.42.1 Detailed Description

Implementation of base classes to access SDH.

11.42.2 General file information

Author

Dirk Osswald

Date

2007-06-13

11.42.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.43 sdh/sdhserial.py File Reference

Implementation of class to access SDH via RS232.

Classes

- class sdh.sdhserial.cSDHSerial

    The class to communicate with a SDH via RS232.

Packages

- package sdh.sdhserial

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::sdhserial.__doc__ = "Class to access SDH via RS232"
- string sdh::sdhserial.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::sdhserial.__url__ = "http://www.schunk.com"
- string sdh::sdhserial.__version__ = "$Id: sdhserial.py 11438 2014-02-28 14:24:55Z Osswald2 $"
- string sdh::sdhserial.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

11.43.1 Detailed Description

Implementation of class to access SDH via RS232.

11.43.2 General file information

Author

Dirk Osswald

Date

2007-06-13

11.43.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.44 sdh/tcpserial.py File Reference

Simple wrapper to access a TCP port like a serial port.

Classes

- class sdh.tcpserial.tTCPSerial
  Simple wrapper class to access a TCP port like a serial port as a file like object.

Packages

- package sdh.tcpserial

11.44.1 Detailed Description

Simple wrapper to access a TCP port like a serial port.

11.44.2 General file information

Author

Dirk Osswald

Date

2010-10-19

11.44.3 Copyright

- Copyright (c) 2010 SCHUNK GmbH & Co. KG

11.45 sdh/tkdsa.py File Reference

Simple tkInter elements to visualize tactile sensors of SDH.

Classes

- class sdh.tkdsa.cSDHTactileSensorPatch

  A class to store a tactile sensor patch.

- class sdh.tkdsa.cTkSDHTactileSensorPatch

  A widget to display a single tactile sensor patch.

- class sdh.tkdsa.cTkSDHTactileSensorPatches

  Widget to display all tactile sensor patches of an SDH.

Packages

- package sdh.tkdsa

Python specific variables

Some definitions that describe the module for python.

- string sdh::tkdsa.__doc__ = "Simple tkInter elements to visualize tactile sensors of SDH"
- string sdh::tkdsa.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::tkdsa.__url__ = "http://www.schunk.com"
- string sdh::tkdsa.__version__ = "$Id: tkdsa.py 4355 2009-05-04 17:17:39Z Osswald2 $"
- string sdh::tkdsa.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"
- tuple sdh::tkdsa.dbg = sdh.dbg.tDBG(False, "cyan")
- def sdh::tkdsa.DisplayStyles

  Return a list of valid display styles.

11.45.1 Detailed Description

Simple tkInter elements to visualize tactile sensors of SDH.

11.45.2 General file information

Author

Dirk Osswald

Date

2007-05-14

11.45.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.46 sdh/unit.py File Reference

Implementation of classes that deal with physical units.

Classes

- class sdh.unit.cUnitConverter

  Unit conversion class.

Packages

- package sdh.unit

Variables

Python specific variables

Some definitions that describe the module for python.

- string sdh::unit.__doc__ = "Unit conversion class and objects."
- string sdh::unit.__author__ = "Dirk Osswald: dirk.osswald@de.schunk.com"
- string sdh::unit.__url__ = "http://www.schunk.com"
- string sdh::unit.__version__ = "$Id: unit.py 4121 2009-02-11 19:30:06Z Osswald2 $"
- string sdh::unit.__copyright__ = "Copyright (c) 2007 SCHUNK GmbH & Co. KG"

Predefined unit conversion objecs

Some predefined cUnitConverter unit conversion objects to convert values between different unit systems. These can be used e.g. to convert angle values between degrees and radians, temperatures between degrees celsius and degrees fahrenheit or the like.

The cSDH class uses such 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.

- tuple sdh::unit.uc_angle_degrees = cUnitConverter("angle", "degrees", u"\N{DEGREE SIGN}", 1.0, 0.0, 1)

  Default converter for angles (internal unit == external unit): degrees.

- tuple sdh::unit.uc_angle_radians = cUnitConverter("angle", "radians", "rad", (2.0∗math.pi)/360.0, 0.0, 3)

  Converter for angles: external unit = radians.

- tuple sdh::unit.uc_time_seconds = cUnitConverter("time", "seconds", "s", 1.0, 0.0, 3)

  Default converter for times (internal unit == external unit): seconds.

- tuple sdh::unit.uc_time_milliseconds = cUnitConverter("time", "milliseconds", "ms", 1000.0, 0.0, 0)

  Converter for times: external unit = milliseconds.

- tuple sdh::unit.uc_temperature_celsius = cUnitConverter("temparature", "degrees celsius", u"\N{DEGREE SIGN}C", 1.0, 0.0, 1)

  Default converter for temparatures (internal unit == external unit): degrees celsius.

- tuple sdh::unit.uc_temperature_fahrenheit = cUnitConverter("temparature", "degrees fahrenheit", u"\N{DEGREE SIGN}F", 1.8, 32.0, 1)

  Converter for temperatures: external unit = degrees fahrenheit.

- tuple sdh::unit.uc_angular_velocity_degrees_per_second = cUnitConverter("angular velocity", "degrees/second", u"\N{DEGREE SIGN}/s", 1.0, 0.0, 2)

  Default converter for angular velocities (internal unit == external unit): degrees / second.

- tuple sdh::unit.uc_angular_velocity_radians_per_second = cUnitConverter("angular velocity", "radians/second", "rad/s", (2.0∗math.pi)/360.0, 0.0, 4)

  Converter for angular velocieties: external unit = radians/second.

- tuple sdh::unit.uc_angular_acceleration_degrees_per_second_squared = cUnitConverter("angular acceleration", "degrees/(second∗second)", u"\N{DEGREE SIGN}/s\N{SUPERSCRIPT TWO}", 1.0, 0.0, 1)

  Default converter for angular velocities (internal unit == external unit): degrees / (second ∗ second)

- tuple sdh::unit.uc_angular_acceleration_radians_per_second_squared = cUnitConverter("angular acceleration", "radians/(second∗second)", u"rad/s\N{SUPERSCRIPT TWO}", (2.0∗math.pi)/360.0, 0.0, 3)

  Converter for angular velocieties: external unit = radians/(second∗second)

- tuple sdh::unit.uc_motor_current_ampere = cUnitConverter("motor current", "Ampere", "A", 1.0, 0.0, 3)

  Default converter for motor current (internal unit == external unit): Ampere.

- tuple sdh::unit.uc_motor_current_milliampere = cUnitConverter("motor current", "milli Ampere", "mA", 1000.0, 0.0, 0)

  Converter for motor current: external unit = milli Ampere.

- tuple sdh::unit.uc_position_millimeter = cUnitConverter("position", "millimeter", "mm", 1.0, 0.0, 1)

  Default converter for position (internal unit == external unit): millimeter.

- tuple sdh::unit.uc_position_meter = cUnitConverter("position", "meter", "m", 1/1000.0, 0.0, 4)

  Converter for position: external unit = meter.

11.46.1 Detailed Description

Implementation of classes that deal with physical units.

11.46.2 General file information

Author

Dirk Osswald

Date

2007-06-13

11.46.3 Copyright

Copyright (c) 2007 SCHUNK GmbH & Co. KG

11.47 sdh/util.py File Reference

Some basic utilities, see also util.__doc__.

Classes

- class sdh.util.tMyOptionParser

  OptionParser with some default options already set: -d | --debug turn on debug (set options.debug flag) -v | --version print version and exit.

Packages

- package sdh.util

Functions

- def sdh::util.GetColor

  return a string that when printed sets the color to c, where c must be in normal, bold, red, green, yellow, blue, magenta, cyan, white, black, for normal color or black_back, red_back, green_back, yellow_back, blue_back, cyan_back, magenta_back, white_-back for reverse color If the environment variable "TERM" is set to "eclipse" then no color string is returned.

- def sdh::util.Beep

  Do n console beeps with a delay of delay seconds.

- def sdh::util.GetClipboard

  Return content of clipboard (on cygwin/Windows)

- def sdh::util.SetClipboard

  Set content of clipboard to content (on cygwin/Windows)

- def sdh::util.WinpathToCygpath

  Return the cygwin path of the file with the windows path winpath.

- def sdh::util.error
- def sdh::util.Ziplen

  return a list containing tuples of elements and indexes of these elements Remark: this is like the std enumerate(l) with the elements in the tuples reversed

- def sdh::util.Call

  call function fun with arguments pars.

- def sdh::util.sgn

  return signum of v

- def sdh::util.GetDefineOrVariable

  Return value of C/C++ define "name" in header file "ifile" or of python variable "name" in python module "ifile".

- def sdh::util.GetProjectName

  Return name of project (extracted from header file release_file).

- def sdh::util.GetProjectRelease

  Return release of project (extracted from header file release_file).

- def sdh::util.RangeDefToList

return a list of indexes according to a range definition string e.g.

- def sdh::util.GetPersistantDict

  Dictionary that stores objects persistently using shelve.

Variables

- string sdh::util.__doc__

  The docstring describing the purpose of the script:

11.47.1 Detailed Description

Some basic utilities, see also util.__doc__.

11.47.2 General file information

Author

Dirk Osswald

Date

2006-04-07

11.47.3 Copyright

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

11.48 sdh/utils.py File Reference

Provide some widely useful utilities. Safe for "from utils import.

Classes

- class sdh.utils.bool

  Introduced in 2.3.

- class sdh.utils.BaseSet

  sets module introduced in 2.3

- class sdh.utils.frozenset
- class sdh.utils.set

- class sdh.utils.DefaultDict

  Dictionary with a default value for unknown keys.

- class sdh.utils.Struct

  Create an instance with argument=value slots.

- class sdh.utils.Queue

  Queue is an abstract class/interface.

- class sdh.utils.FIFOQueue

  A First-In-First-Out Queue.

- class sdh.utils.PriorityQueue

  A queue in which the minimum (or maximum) element (as determined by f and order) is returned first.

Packages

- package sdh.utils

Functions

- def sdh::utils.sum

  Introduced in 2.3.

- def sdh::utils.enumerate

  Return an iterator that enumerates pairs of (i, c[i]).

- def sdh::utils.reversed

  Iterate over x in reverse order.

- def sdh::utils.sorted

  Copy seq and sort and return it.

- def sdh::utils.Dict

  Create a dict out of the argument=value arguments.

- def sdh::utils.update

  Update a dict; or an object with slots; according to entries.

- def sdh::utils.removeall

  Return a copy of seq (or string) with all occurences of item removed.

- def sdh::utils.unique

Remove duplicate elements from seq.

- def sdh::utils.product

  Return the product of the numbers.

- def sdh::utils.count_if

  Count the number of elements of seq for which the predicate is true.

- def sdh::utils.find_if

  If there is an element of seq that satisfies predicate; return it.

- def sdh::utils.every

  True if every element of seq satisfies predicate.

- def sdh::utils.some

  If some element x of seq satisfies predicate(x), return predicate(x).

- def sdh::utils.isin

  Like (elt in seq), but compares with is, not ==.

- def sdh::utils.argmin

  Return an element with lowest fn(seq[i]) score; tie goes to first one.

- def sdh::utils.argmin_list

  Return a list of elements of seq[i] with the lowest fn(seq[i]) scores.

- def sdh::utils.argmin_random_tie

  Return an element with lowest fn(seq[i]) score; break ties at random.

- def sdh::utils.argmax

  Return an element with highest fn(seq[i]) score; tie goes to first one.

- def sdh::utils.argmax_list

  Return a list of elements of seq[i] with the highest fn(seq[i]) scores.

- def sdh::utils.argmax_random_tie
- def sdh::utils.histogram

  Return a list of (value, count) pairs, summarizing the input values.

- def sdh::utils.log2

  Base 2 logarithm.

- def sdh::utils.mode

  Return the most common value in the list of values.

- def sdh::utils.median

Return the middle value, when the values are sorted.

- def sdh::utils.mean

  Return the arithmetic average of the values.

- def sdh::utils.stddev

  The standard deviation of a set of values.

- def sdh::utils.dotproduct

  Return the sum of the element-wise product of vectors x and y.

- def sdh::utils.vector_add

  Component-wise addition of two vectors.

- def sdh::utils.probability
- def sdh::utils.num_or_str

  The argument is a string; convert to a number if possible, or strip it.

- def sdh::utils.normalize

  Multiply each number by a constant such that the sum is 1.0 (or total).

- def sdh::utils.turn_right
- def sdh::utils.turn_left
- def sdh::utils.distance
- def sdh::utils.distance2
- def sdh::utils.clip

  Return vector, except if any element is less than the corresponding value of lowest or more than the corresponding value of highest, clip to those values.

- def sdh::utils.printf

  Format args with the first argument as format string, and write.

- def sdh::utils.caller

  Return the name of the calling function n levels up in the frame stack.

- def sdh::utils.memoize

  Memoize fn: make it remember the computed value for any argument list.

- def sdh::utils.if_

  Like C++ and Java's (test ? result : alternative), except both result and alternative are always evaluated.

- def sdh::utils.name
- def sdh::utils.isnumber
- def sdh::utils.issequence
- def sdh::utils.print_table

Print a list of lists as a table, so that columns line up nicely.

- def sdh::utils.Stack

  Return an empty list, suitable as a Last-In-First-Out Queue.

Variables

- float sdh::utils.infinity = 1.0e400
- list sdh::utils.orientations = [(1,0), (0, 1), (-1, 0), (0, -1)]

  OK, the following are not as widely useful utilities as some of the other functions here, but they do show up wherever we have 2D grids: Wumpus and Vacuum worlds, TicTacToe and Checkers, and markov decision Processes.

- dictionary sdh::utils.Fig = { }

  Fig: The idea is we can define things like Fig[3,10] later.

11.48.1 Detailed Description

Provide some widely useful utilities. Safe for "from utils import.

11.48.2 General file information

Author

Russell and Norvig's 'Artificial Intelligence: A Modern Approach', see e.g. `http://code.google.co`

Date

2006-04-07

11.49 sdhlibrary_python.dox File Reference

11.50 setup.py File Reference

Python distutils setup script for sdh package.

Packages

- package setup

Functions

- def setup.Pathify
  join dirs (list of directory names/files) to a list of paths using the right path separator

Variables

- tuple setup.sdh_locals = dict()
- tuple setup.sdh_globals = dict()
- list setup.doc_files
- list setup.guidat_files
- tuple setup.src_rel_paths = map(lambda n: Pathify(r, n)[0], f)
- tuple setup.target_path
- list setup.version = sdh_locals["PROJECT_RELEASE"]
- string setup.description = 'sdh: the python package to access an SDH (SCHUNK Dexterous Hand)'
- string setup.author = 'Dirk Osswald'
- string setup.author_email = 'dirk.osswald@de.schunk.com'
- string setup.url = 'http://www.schunk.com/'
- string setup.long_description
- list setup.packages = ['sdh']
- tuple setup.scripts = Pathify('demo', 'demo-simple.py')
- setup.data_files = doc_files+guidat_files

11.50.1 Detailed Description

Python distutils setup script for sdh package.

11.50.2 General file information

Author

Dirk Osswald

Date

2007-05-11

11.50.3 Copyright

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

Index