

## SDH2 Configuration and Update

This documentation describes the following features of the SDH2:

- Communication
- Configuration commands
- Updating the firmware

### 1. Communication

The following section is just a technical note, if you access the SDH2 from a PC using the provided SDHLibrary then the functional interface will be the same regardless whether you use Ethernet (TCP/IP), CAN bus or RS232.

In the SDH2 Ethernet (TCP/IP), CAN bus or RS232 can be used for communication with the hand. But the communication data – the protocol – is the same ASCII- or binary-stream based one for all physical communication interfaces.

For ASCII based commands each command must be terminated by a “;” or “\r\n” (0x0d 0x0a) sequence, even when sent via Ethernet (TCP/IP) or CAN, the same is true for the replies from the SDH.

#### 1.1. RS232

RS232 is per se a stream based communication. Commands (requests) to the SDH2 are received on the Rx input of the RS232 interface. Likewise answers (responses) from the SDH2 are sent via the Tx output of the RS232 interface.

This allows for simple testing of the SDH2 RS232 communication without SDH2 specific software on the PC. Just start a standard terminal client like Windows `HyperTerminal` or Linux `minicom` and connect to the RS232 interface where the SDH is connected and you can type in commands and receive answers.

#### 1.2. CAN bus

CAN bus provides a frame based communication. A frame is a small packet of up to 8 data bytes. A so called CAN-ID is part of each CAN frame. Different IDs can be used to distinguish different types of data, senders or receivers according to some higher layer protocol. The approach taken by the SDH2 is to mimic a stream based communication by using only two configurable CAN-IDs, one for reading and one for writing. This way the SDH2 can easily be integrated into a CAN-bus using other higher layer protocols.

The commands (requests) to the SDH2 are received simply by accumulating all the data bytes with the CAN messages for the configurable read CAN-ID. Likewise the answers (responses) from the SDH2 are sent as the data bytes of the CAN messages with the other configurable write CAN-ID.

#### 1.3. Ethernet (TCP/IP)

new in firmware 0.0.3.1

TCP communication on a specific TCP port is per se a stream based communication. Commands (requests) to the SDH2 are received on the input of the port. Likewise answers (responses) from the SDH2 are sent via the output of the port.

The default port (23) is the well-known telnet port. This allows for simple testing of the SDH2 TCP/IP communication without SDH2 specific software on the PC. Just start a standard telnet client and connect to the IP of the SDH and you can type in commands and receive answers.

The SDH2 now also contains a simple http web server. This can be used to configure or operate the SDH with a standard web browser. Even simple operations of the SDH2, like interactively moving the fingers are possible. JavaScript must be enabled in the browser to make this work. The http web server requires additional data (filesystem.flash) installed into the flash memory of the SDH2, see section 4.3.1.

new in firmware 0.0.3.2

The tactile sensor controller (DSACON32m) can be accessed via TCP as well. The native binary DSACON32m protocol is simply streamed back and forth on a TCP connection using TCP-port 1300 by default

## 2. Configuration commands

Some communication parameters can now be configured ‘in the field’, i.e. by you, the User. This configuration can be done by any RS232 terminal program. The configuration commands given below are understood by the High Level Controller (HLC) during normal operation, i.e. via RS232, CAN bus, or Ethernet TCP/IP. Since not many users will have a terminal program that operates via CAN the configuration commands are also understood by the Low Level Control configuration utility, which will always operate via RS232. To enter that utility open a RS232 terminal program like Windows HyperTerminal or Linux minicom (115200 bit/s, 8 data bits, no parity, 1 stop bit). Then reset or restart the SDH. The SDH will detect the connected terminal and will check if the space key + Return key is pressed within 3 seconds. If so, then the configuration utility is run. Try typing help to get a list of commands or read on.

The configuration commands mentioned below are also available in the RemoteUpdate configuration (see section 4.1.2). To enter that environment open a RS232 terminal program like Windows HyperTerminal or Linux minicom (115200 bit/s, 8 data bits, no parity, 1 stop bit). Then switch on the SDH and press the reset button for more than 10 s (until the blue LED is switched off, see also sections 4.2.1 “LED and reset button placement and naming” and 4.2.6 “RemoteUpdate mode”). Try typing help to get a list of commands or read on. Attention: in the RemoteUpdate environment there must be at least one space between the command name and the “=” character. That is e.g. “cfg\_can\_baudrate =500000” and not “cfg\_can\_baudrate=500000”

new in firmware 0.0.1.7

### 2.1. cfg\_print

The command „cfg\_print“ prints out the current configuration. In a terminal this looks like:

```
cfg_print
```

Response from SDH:

```
valid:.....1
hw_revision:.....3
serial_nr:.....27
absenc_offset[0]:.....179.296875 [deg]
absenc_offset[1]:.....294.960938 [deg]
absenc_offset[2]:.....125.156250 [deg]
absenc_offset[3]:.....240.117188 [deg]
absenc_offset[4]:.....291.796875 [deg]
absenc_offset[5]:.....118.125000 [deg]
absenc_offset[6]:.....180.351562 [deg]
config_data_revision:.6
comm_interface:.....0 [0=RS232, 1=CAN, 2=TCP (telnet)]
can_baudrate:.....500000 [bit/s]
can_id_read:.....42 [= 0x02a]
can_id_write:.....43 [= 0x02b]
rs232_baudrate:.....115200 [bit/s]
rs232_channel_dsa:.....0
net_mac_id:.....02-07-ed-00-00-1b
net_ip:.....192.168.1.42
net_mask:.....255.255.255.0
net_gw:.....192.168.1.1
net_telnet_port:.....23 [0=off]
net_http_port:.....80 [0=off]
pid[0]:.....25.000000,4000.000000,0.000000
pid[1]:.....20.000000,4000.000000,0.000000
pid[2]:.....45.000000,5000.000000,0.000000
pid[3]:.....20.000000,4000.000000,0.000000
pid[4]:.....50.000000,5000.000000,0.000000
pid[5]:.....20.000000,4000.000000,0.000000
pid[6]:.....50.000000,5000.000000,0.000000
```



Only the parameters described below can be configured. The values shown above are the default values. (The default Ethernet MAC-ID will use the serial\_nr of the SDH2 as its last byte to individualize the ID). Any change done is stored in non volatile EEPROM memory and will be used on the next boot (reset or power on).

When you update the firmware to a newer version with additional configurable parameters the “old” parameter values will be retained while “new” parameter values will be set to their default value.

## 2.2. **cfg\_default**

Resets all values of the configuration to their default.

```
cfg_default
```

See `cfg_print` above to find out which are the default values.

## 2.3. **cfg\_comm\_interface**

Without parameters the command „`cfg_comm_interface`“ just prints a code for the current communication interface. With a parameter the interface can be changed. Code 0 is for RS232, code 1 for CAN and code 2 for Ethernet (TCP/IP).

Example: to switch the interface to CAN use:

```
cfg_comm_interface=1
```

## 2.4. **cfg\_can\_baudrate**

Without parameters the command „`cfg_can_baudrate`“ just prints the current CAN baudrate in bit/s. With a parameter the baudrate can be changed. Allowed values are 1000000, 500000, 250000, 100000

Example: to change the baudrate to 500 kBit:

```
cfg_can_baudrate=500000
```

## 2.5. **cfg\_can\_id\_read**

Without parameters the command „`cfg_can_id_read`“ just prints the current CAN ID used for reading.

With a parameter the ID can be changed. Allowed values are 0-2047. The value must be given in decimal

Example: to change the ID to 24 use:

```
cfg_can_id_read=24
```

## 2.6. **cfg\_can\_id\_write**

Without parameters the command „`cfg_can_id_read`“ just prints the current CAN ID used for reading.

With a parameter the ID can be changed. Allowed values are 0-2047. The value must be given in decimal

Example: to change the ID to 24 use:

```
cfg_can_id_read=24
```

## 2.7. **cfg\_rs232\_baudrate**

The RS232 baudrate can be changed too. The default is 115200 bit/s. The baudrate for the High Level Controller (HLC) can be configured within 110 and 921600 bit/s. The HLC communicates with the outside world during normal operation, i.e. after start.

The Low Level Controller (LLC) always uses the default baudrate 115200 bit/s. The LLC only communicates with the outside world during start-up, see the description of the configuration utility in section 2 above.

Example: to change the RS232 baudrate to 38400 bit/s use:

```
cfg_rs232_baudrate=38400
```

## 2.8. **cfg\_rs232\_channel\_dsa**

The RS232 channel to be used by the tactile sensor controller (DSACON32m) can be configured. There are two RS232 channels available, channel 0 and 1.

new in firmware 0.0.1.6

new in firmware 0.0.1.10



- If the SDH is configured to use RS232 for movement commands (see section 2.3) then the first RS232 channel (channel 0) is normally used for these movements commands and the second channel (channel 1) is used for the tactile sensor data, no matter what channel number is configured with the `cfg_rs232_channel_dsa` command.
- If the SDH is configured to use CAN for movement commands (see section 2.3) then the channel for the tactile sensor data is determined by the setting of the `cfg_rs232_channel_dsa` command. The default is to use channel 0.

Example: to use the second channel 1 for the tactile sensor data, even if CAN is used for movement commands use:

```
cfg_rs232_channel_dsa=1
```

## 2.9. `cfg_sn`

new in firmware 0.0.2.4

The serial number can be configured. For setting the value you must enter the “root” password, but getting the value can be done by anyone

Example: to read the serial number only use:

```
cfg_sn
```

## 2.10. `cfg_hw_revision`

new in firmware 0.0.2.4

The hardware revision number can be configured. For setting the value you must enter the “root” password, but getting the value can be done by anyone

Example: to read the hardware revision only use:

```
cfg_hw_revision
```

## 2.11. `cfg_pid`

new in firmware 0.0.2.8

The PID parameters of the axis controllers can be configured.

**WARNING:** Changing the PID parameters affects the control behaviour of the axis controllers. Setting inadequate values might make the axis move unexpectedly or it might even damage the mechanics if such parameters are used for a longer time.

Example: to read the PID parameters of axis 0 use:

```
cfg_pid(0)
```

which will reply something like:

```
CFG_PID(0)=25.000000,4000.000000,0.000000
```

In order to change the values use `cfg_pid(AXIS_INDEX)=PVALUE, IVALUE, DVALUE` Here is an example for setting axis 0:

```
cfg_pid(0)=25,4000,0
```

and the hand will reply with the values actually set:

```
CFG_PID(0)=25.000000,4000.000000,0.000000
```

valid values for P are [0.0 .. 1024.0] with a resolution of 0.5

valid values for I are [0.0 .. 1024000.0] with a resolution of 500.0

valid values for D are [0.0 .. 1.024] with a resolution of 0.0005

**Hint:** The parameters set with `cfg_pid` are stored persistently and thus will remain active, even after a reset or power cycle. For experimenting with the PID parameters these can be accessed from the SDHLibrary and are then stored only temporarily. The demo-gui.py Python application allows to get/set the PID parameters as well (since SDHLibrary-Python 0.0.1.18 from 2010-02-02), see menu Debug → PID adjust.

## 2.12. `cfg_net_mac_id`

new in firmware 0.0.3.1

Without parameters the command „`cfg_net_mac_id`“ just prints the current Ethernet MCA-ID used by the Ethernet interface. With a parameter the ID can be changed by user root. The value must be given as 6 two digit hexadecimal numbers separated by “-”.

Example: to change the ID to 00-01-02-03-04-05 use:

```
cfg_net_mac_id=00-01-02-03-04-05
```

### 2.13. **cfg\_net\_ip**

new in firmware 0.0.3.1

Without parameters the command „cfg\_net\_ip“ just prints the current IP address of the SDH used by the TCP/IP interface. With a parameter the IP can be changed. The value must be given in canonical form as 4 decimal numbers separated by “.”.

Example: to change the IP to 192.168.100.2 use:

```
cfg_net_ip=192.168.100.2
```

### 2.14. **cfg\_net\_mask**

new in firmware 0.0.3.1

Without parameters the command „cfg\_net\_ip“ just prints the current IP mask of the SDH used by the TCP/IP interface. With a parameter the mask can be changed. The value must be given in canonical form as 4 decimal numbers separated by “.”.

Example: to change the mask to 255.255.0.0 use:

```
cfg_net_ip=255.255.0.0
```

### 2.15. **cfg\_net\_gw**

new in firmware 0.0.3.1

Without parameters the command „cfg\_net\_gw“ just prints the current gateway address of the SDH used by the TCP/IP interface. With a parameter the gateway can be changed. The value must be given in canonical form as 4 decimal numbers separated by “.”.

Example: to change the gateway to 192.168.100.254 use:

```
cfg_net_ip=192.168.100.254
```

### 2.16. **cfg\_net\_telnet\_port**

new in firmware 0.0.3.1

Without parameters the command „cfg\_net\_telnet\_port“ just prints the current telnet port number of the SDH used by the TCP/IP interface. This port is not only used for TCP communication via telnet, but also for TCP communication by the SDHLibrary. With a parameter the port number can be changed. The value must be given as a decimal number.

Example: to change the port to 50023 use:

```
cfg_net_telnet_port=50023
```

Setting the http port to 0 (zero) will disable the telnet server.

### 2.17. **cfg\_net\_http\_port**

new in firmware 0.0.3.1

Without parameters the command „cfg\_net\_http\_port“ just prints the current http port number of the SDH used by the TCP/IP interface. This port is used for the simple http web server integrated into the SDH. With a parameter the port number can be changed. The value must be given as a decimal number.

Example: to change the port to 8080 use:

```
cfg_net_http_port=8080
```

Setting the http port to 0 (zero) will disable the http server.

### 2.18. **cfg\_net\_dsa\_port**

new in firmware 0.0.3.2

Without parameters the command „cfg\_net\_dsa\_port“ just prints the current TCP port number of the SDH used for the tactile sensor data. This port is used for simple forwarding of the binary DSA protocol data to and from the tactile sensor controller (DSACON32m). With a parameter the port number can be changed. The value must be given as a decimal number. The default is 13000.

Example: to change the port to 13001 use:

```
cfg_net_http_port=13001
```

Setting the DSA port to 0 (zero) will disable the DSA TCP server.

### 3. Testing/Debugging commands

#### 3.1. change\_rs232

This command can be used to switch the RS232 connection temporarily to the tactile sensor controller (DSACON32m). This command is not really a configuration command as the setting is not stored in the EEPROM and thus the effect will be lost upon the next reboot.

Example: to change the RS232 connection to the tactile sensor use:

```
change_rs232=3
```

#### 3.2. change\_channel

This command can be used to switch the communication channel temporarily (until next change\_channel command or until reset/restart). This command is not really a configuration command as the setting is not stored in the EEPROM and thus the effect will be lost upon the next reboot. Code 0 is for RS232 and code 1 for CAN.

Example: to temporarily switch the interface to CAN use:

```
change_channel=1
```

Remark: This command will immediately switch the communication channel. Thus no response will be replied for the command itself on the currently used communication channel and no further commands will be accepted on this channel.

#### 3.3. hw\_test

new in firmware 0.0.2.5

Strictly speaking this command is not a configuration command. Instead it is for testing of some of the hardware aspects of an SDH. But the command is available in the configuration utility of the low level controller only. To start displaying some hardware info enter:

```
hw_test
```

This will cyclically display a screen like the following:

```
SDH2 low level interactive hardware test.
```

```
The test runs continuously, until you press <x>.
```

	proximal joint			distal joint			
Fx	inc-encoder	abs-encoder	temp	inc-encoder	abs-encoder	temp	Status
F0	-1( )	+220.08(+)	26.8				incomplete
F1	+0( )	-40.78( )	22.8	+0( )	-307.62( )	22.5	incomplete
F2	+3246( )	+116.37( )	23.0	+22758( )	+45.35( )	22.5	incomplete
F3	+1( )	-152.23( -)	failed	+0( )	-101.25( -)	failed	error(s)

```

FPGA-temperature:  37.5
PCB-temperature :  34.0

```

The output shows the finger number in column 1. Then the current values of the incremental and absolute encoders (plus some additional info described below) and the motor temperature is shown for the root or proximal joints in columns 2, 3 and 4. For the fingers 1-3 corresponding information is shown for the distal joints in columns 5-7. The last status column shows the test status of the finger, which can be “incomplete”, “error(s)” or “OK”.

An error status indicates that some hardware cannot be accessed. In the example output above the temperature sensors of F3 could not be read (since no finger 3 was connected). If there are no such errors then the status will remain “incomplete” until you manually move the corresponding joints in both directions by a specific angle. The status will change to “OK” if joint angle changes are detected in positive as well as in negative direction for both the incremental and the absolute encoders of a joint. These change detections are indicated by the (+), (-) and finally (+-) indicators after the numeric values in the columns.

## 4. Updating the firmware

This chapter explains in detail what and how to update the firmware of the SDH2. Section 4.1 explains the term firmware in the context of the SDH2, section 4.2 describes how to enter the various modes to update parts of the firmware and section 4.3 describes how to actually update the firmware. At the end of section 4.3 there is a short summary of the steps needed.

### 4.1. Preliminary remarks

#### 4.1.1. Firmware

The SDH2 has an FPGA as main processing unit. Within the FPGA (soft-) processors are used as CPUs to execute the control software of the system. The FPGA has to be configured on start up. Such an FPGA configuration is stored in non volatile flash memory and is loaded on power up or reset. After that the control software, which is also stored in same flash memory, is executed. Therefore the firmware for the SDH2 is threefold:

- an FPGA configuration
- the control software that runs on the (soft-)processors of the FPGA-configuration
- a file system that contains the http web server data

If the SDH2 is equipped with tactile sensors then it includes an additional sensor controller (DSACON32m). This controller also has a processor and control software. Therefore then the firmware of the SDH2 has a third component:

- the control software for the tactile sensor processor

#### 4.1.2. Configurations

The SDH2 actually has stored two different FPGA configurations and corresponding control software within its flash memory:

- The “normal” configuration with two processors for high- and low-level control (HLC/LLC)
- A “RemoteUpdate” configuration for testing and for updating the firmware of the SDH2

This allows the customer to safely update the firmware of the SDH2 within his system without specialised programming hardware like JTAG adapters.

### 4.2. Reset button and boot control

After power-on the „normal“ FPGA configuration and control software is loaded and executed. This enables normal operation of the hand right from the start with no further intervention.

To choose specific boot modes, e.g. for updating the firmware, the reset button is used. The reset button of the SDH2 is normally hidden under a sealed screw besides the LEDs of the SDH2. To activate one of the boot modes described below unscrew the screw with a 2mm Allen wrench. The reset button can then be pressed like described below with a thin bolt, e.g. the just used Allen wrench.

#### 4.2.1. LED and reset button placement and naming

The SDH device has 5 built in LEDs (Light Emitting Diodes). These are used to indicate various internal states. Some LEDs are used to display different types of information, depending on the current mode of the hand. Figure 1 shows the placement and the naming convention of the reset button and the LEDs.

new in firmware 0.0.3.1





**Figure 1:** LED placement and naming:

- /1/ - Reset button (normally hidden behind a screw)
- /2/ - Power LED (blue)
- /3/ - Busy LED (orange)
- /4/ - Error LED (red)
- /5/ - Ethernet Traffic LED (yellow)
- /6/ - Ethernet Link LED (green)

#### 4.2.2. Boot modes

Unfortunately several different boot modes must be available in order to be able to update specific parts of the SDH2 firmware. Therefore the length of pressing the reset button is used to determine which boot mode to activate. To determine which boot mode will be selected some blink codes of the blue power LED of the SDH2 are used. The blink codes are described below. The blinking is indicated even before the reset button is released, so as soon as you see the desired blink code of the blue power LED you can release the reset button and the desired boot mode is then selected.

#### 4.2.3. Normal reset

A short press (duration < 1 s) is used as a "normal" reset, so the SDH2 will then behave in the same way as after power on. I.E. the FPGA will be reconfigured with high- and low-level controllers and the corresponding control software will start up into normal operation. The blue power LED will not blink within the first second of pressing the reset button.

#### 4.2.4. DSAICON32m pass-through mode

A longer button press between 1 s and 2 s will activate "DSAICON32m pass-through mode", i.e. the external serial RS232 connection will be routed to the DSAICON32m controller directly, no matter how the communication connections are configured. This can be used to test the tactile sensors, e.g. with the DSA\_Explorer PC-software, independently of the configured communication connections. The FPGA will not be reconfigured in this mode. To indicate that "DSAICON32m pass-through mode" is selected the blue power LED of the SDH will blink once shortly.

#### 4.2.5. DSAICON32m flash-load mode

A longer button press between 2 s and 10 s will activate "DSAICON32m flash-load mode", i.e. the external serial RS232 connection will be routed to the DSAICON32m controller and that controller will be forced into

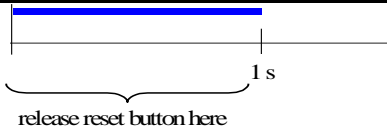
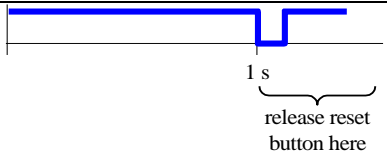
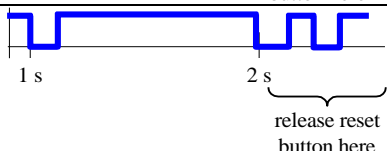
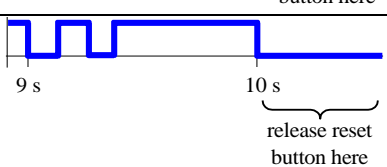


flash-loading mode. The FPGA will not be reconfigured in this mode. To indicate that "DSACON32m flash-load mode" is selected the blue power LED of the SDH will blink twice shortly. How to actually update the DSACON32m control software see section 4.3.3.

#### 4.2.6. RemoteUpdate mode

A very long press over 10 s will reconfigure the FPGA with the RemoteUpdate configuration, i.e. the external serial RS232 connection will be connected to a shell-like control program. To indicate that "RemoteUpdate mode" is selected the blue power LED of the SDH will be switched off. How to actually update the SDH2 FPGA configuration or control software see section 4.3.1.

**Table 1 : Summary of the boot mode selection**

reset button press duration t	boot mode	FPGA configuration & control software	RS232 connection	blink code
$t < 1 \text{ s}$	normal	Normal	as configured (HLC or DSACON32m)	
$1 \text{ s} \leq t < 2 \text{ s}$	DSACON32m pass-through mode	Normal	DSACON32m	
$2 \text{ s} \leq t < 10 \text{ s}$	DSACON32m flash-load mode	Normal	DSACON32m	
$10 \text{ s} \leq t$	RemoteUpdate mode	RemoteUpdate	RemoteUpdate	

### 4.3. Updating firmware

#### 4.3.1. Updating FPGA configuration or control software

Updates for the SDH2 are provided in the form of so called "flash" files for each part of the firmware:

- soc.flash (FPGA configuration)
- hlc.flash (control system high-level)
- llc.flash (control system low-level)
- filesystem.flash (file system for http web server)

Depending on what has actually changed only hlc.flash and/or llc.flash can be updated. But if you change the FPGA configuration, i.e. update soc.flash, then you must also update hlc.flash and llc.flash. The high- and low-level controller software checks on start up if it is running on the right FPGA configuration and stops if not.

Firmware 0.0.3.2 requires a new FPGA configuration, so you have to update soc.flash as well as hlc.flash, llc.flash and filesystem.flash.

To update the FPGA configuration or control software you must first activate RemoteUpdate mode as described in section 4.2.6. The serial RS232 communication is then routed to a shell like program which can be used to interactively test the SDH2. You can connect any terminal program like Windows HyperTerminal or Linux miniterm and issue commands by typing their names with the keyboard. E.g. type "help" + Return for a list of available commands.

Updating SDH2 firmware could be done interactively but that would be extremely time consuming, so a Python script (sdhflash.py) is provided to do the job. The script first uploads a given flash file to the volatile RAM memory of the SDH2. After that the data is stored in non-volatile flash memory of the SDH2.

new in firmware 0.0.3.1

new in firmware 0.0.3.2



The flash file actually is in Motorola SREC format, which is an ASCII based non binary format. On the one hand this enlarges the files to be transferred and thus lengthens the time needed for the upload (several minutes). But on the other hand due to the included checksums the transfer is also very safe and fault-tolerant. An erroneous file will not be accepted.

A windows installer `sdhflash-w.x.y.z.win32.exe` can be provided which contains the actual `sdhflash.py` script, the flash files and this document. The installer will also create shortcuts in the start menu in `Start → SCHUNK → SDH → dhflash-w.x.y.z_DATE` to simplify the calling of the script.

new in firmware 0.0.1.6

#### 4.3.2. Summary

This summarises the steps needed to update SDH2 FPGA configuration or control software. Please repeat the steps for each file that must be updated (usually `hlc.flash`, `llc.flash` and `filesys.flash`):

**Step 1:** Connect the serial RS232 communication of the SDH2 (RxD and TxD pins) to a RS232 port of a PC

**Step 2:** Switch on the SDH2

**Step 3:** Activate RemoteUpdate mode: press the reset button for more than 10 s until the blue power LED is switched off permanently

**Step 4:** On the PC connected to the SDH start the flash process by calling the `sdhflash.py` script with the file to flash as parameter. The script accepts various other command line parameters, see the online help ("`sdhflash.py -h`") for descriptions. For example to send the file `hlc.flash` on the first RS232 port (`/dev/ttyS0` on Unix respectively `COM1` on Windows) use:

```
sdhflash.py -p 0 hlc.flash
```

or use the shortcut from the windows installer:

```
Start → SCHUNK → SDH → sdhflash-w.x.y.z_DATE → sdhflash-HLC
```

You need not provide the RS232 interface with `-p X`, if not provided then the `sdhflash.py` will search for SDHs in RemoteUpdate mode on all RS232 interfaces it can find.

new in firmware 0.0.2.2

**Step 5:** The script then shows some information about the file to flash and gives a rough estimation of the time needed for the upload. The ongoing upload is indicated by printed dots. Every dot represents 1 percent of the file.

**Step 6:** After the file is uploaded the flashing will start. This is displayed by the script and indicated by a single beep. The SDH will show the red LED switched on while flashing. In this phase the power **MUST NOT** be interrupted to prevent data corruption.

**Step 7:** After the file has been successfully flashed the script exits after printing appropriate messages and beeping for 2 times. You can now flash another file or restart the SDH (either by reset or power-off, power-on).

**WARNING:** if you update the RemoteUpdate feature itself then you **MUST** update the SoC (FPGA configuration) and the software **WITHOUT** resetting or restarting the SDH in between. I.E. you must call `sdhflash-SoC-RU` and `sdhflash-RU` right after each other. If you fail to do so then the RemoteUpdate feature will not work anymore!

Depending on the system you are running on, you might have to adjust parameters of the `sdhflash.py` tool in step 4 above. If the tool does not respond as described in steps 5, 6 then try to increase the `maxtries` parameter. Adjust the command line used in step 4 like this:

```
sdhflash.py -p 0 -maxtries=10000 hlc.flash
```

Maybe even larger values might be necessary.

#### 4.3.3. Updating DSA CON32m control software

To update the firmware (control software) of the tactile sensor controller DSA CON32m the 3<sup>rd</sup> party Windows™ tool "Philips™ LPC2000 Flash Utility" is needed.

**Step 1:** Get the Philips flash tool, which is provided by SCHUNK or can be downloaded from the Internet, e.g. via [http://www.nxp.com/products/microcontrollers/support/software\\_download/lpc2000/](http://www.nxp.com/products/microcontrollers/support/software_download/lpc2000/).

**Step 2:** Unzip and install the "[LPC2000\\_flash\\_utility.zip](#)" on a Windows™ PC

**Step 3:** Prepare the SDH:

Connect the SDH to the Windows™ PC, e.g. using the wooden test stand. For updating the first serial connection (channel 0) is needed.

(Remark: You **MUST** use the first RS232 channel (channel 0), even if you normally access the tactile sensors via its second RS232 channel (channel 1).)

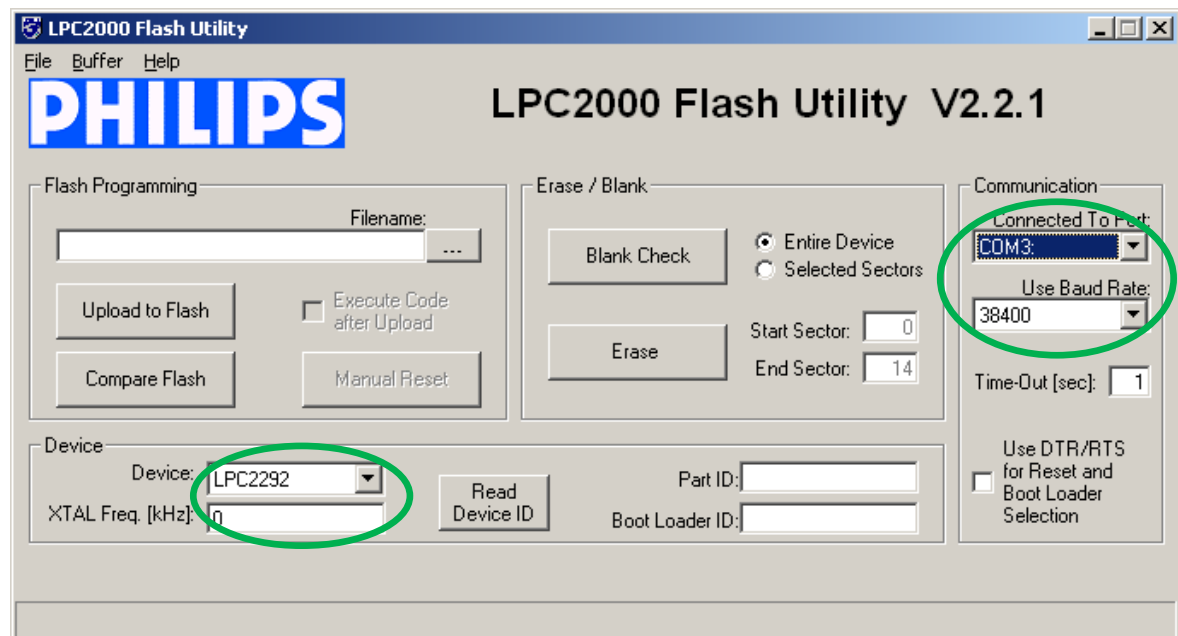
**Step 4:** Power up the SDH**Step 5:** Prepare to determine the hardware revision of the DSACON32m

Unfortunately there exist two different hardware revisions of the tactile sensor controller DSACON32m. These differ in the microcontroller and the quartz clock. The used microcontroller is called “Device” below and in the Philips flash tool and the quartz clock is called “XTAL Freq.”.

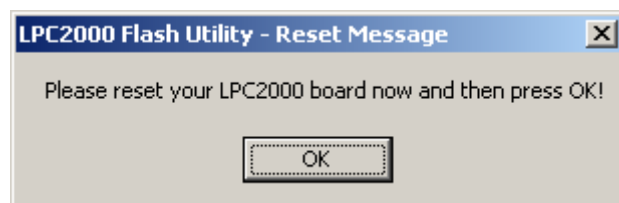
One revision uses “Device” **LPC2292** and “XTAL Freq.” **147456 kHz**, the other one used **LPC2294** and **12000 kHz**. **Unfortunately these different revisions require different firmware versions.**

To determine which hardware revision you have please start the Philips flash utility and enter the following data (see Figure 2):

- Communication:
  - “Connected To Port”: Select the RS232 COM interface which is connected to the RS232 channel **0** of the SDH. E.g.: **COM3**
  - “Use Baud rate”: **38400**
  - “Time-Out [sec]”: **1**
  - Deselect the “USE DTR/RTS” button
- Device:
  - “Device”: **LPC2292**
  - “XTAL Freq. (kHz)”: **0**



**Figure 2:** Tool settings for check the hardware revision of the DSACON32m controller

**Step 6:** Press the “Read Device ID” Button. This will show a pop up window, see Figure 3.

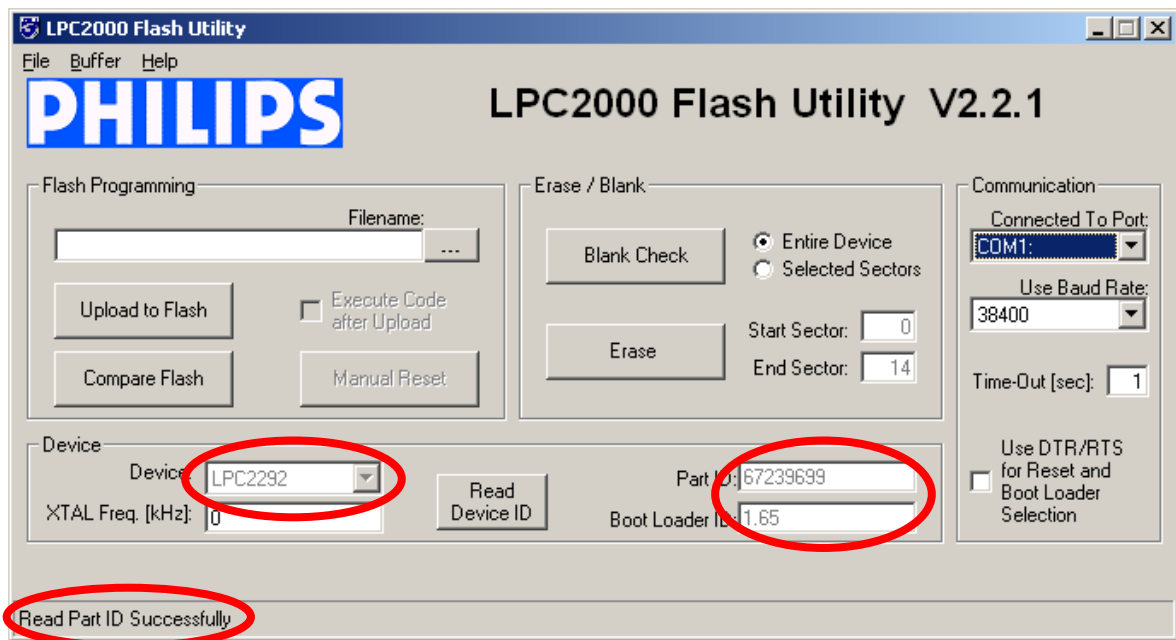
**Figure 3:** Tool Pop up window.

**Step 7:** Before pressing the “OK” button in the pop up window, please press the reset button of the SDH for at least two seconds until the blue LED of the SDH starts blinking twice, see Table 1. This will activate DSA32m flash-load mode as described in section 4.2.5. After that you can press the “OK” button in the pop up window.

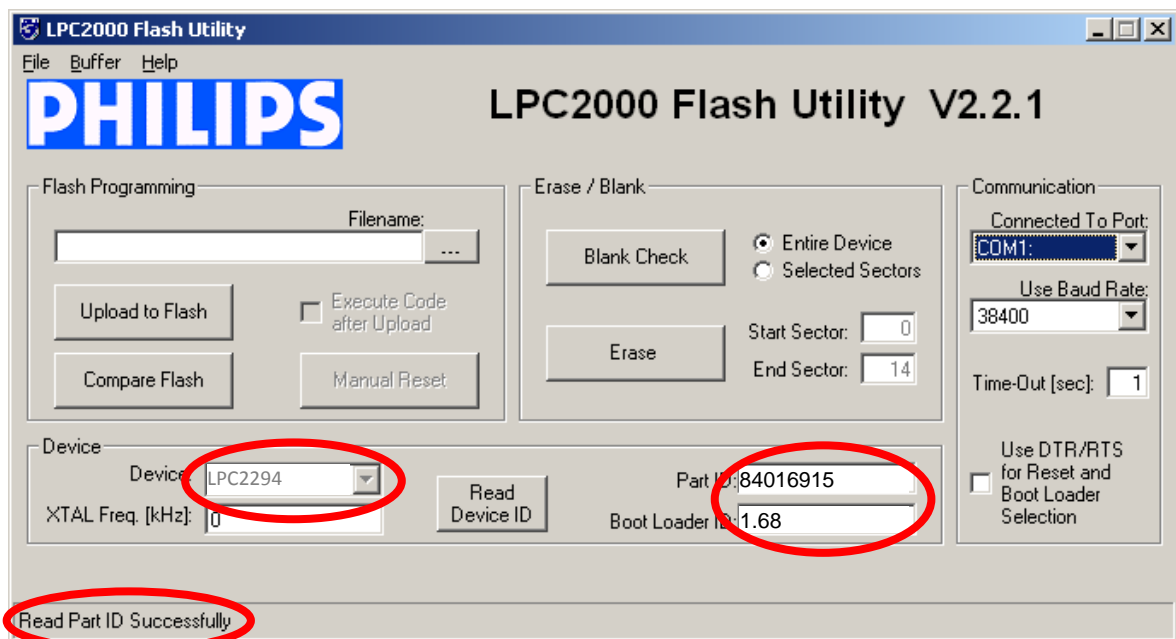
**Step 8:** If the tool could correctly establish a connection to the bootloader of the DSA32m, then the tool window should state “Read Part ID Successfully” in its status line and update its “Device”, “Part ID” and “Boot loader ID” fields.

A DSA32m controller with Device **LPC2292** will keep that entry in the Device field, see Figure 4.

A DSA32m controller with Device **LPC2294** will change that entry in the Device field, see Figure 5.



**Figure 4:** Response from an DSA32m controller with Device **LPC2292**, 14740 kHz.



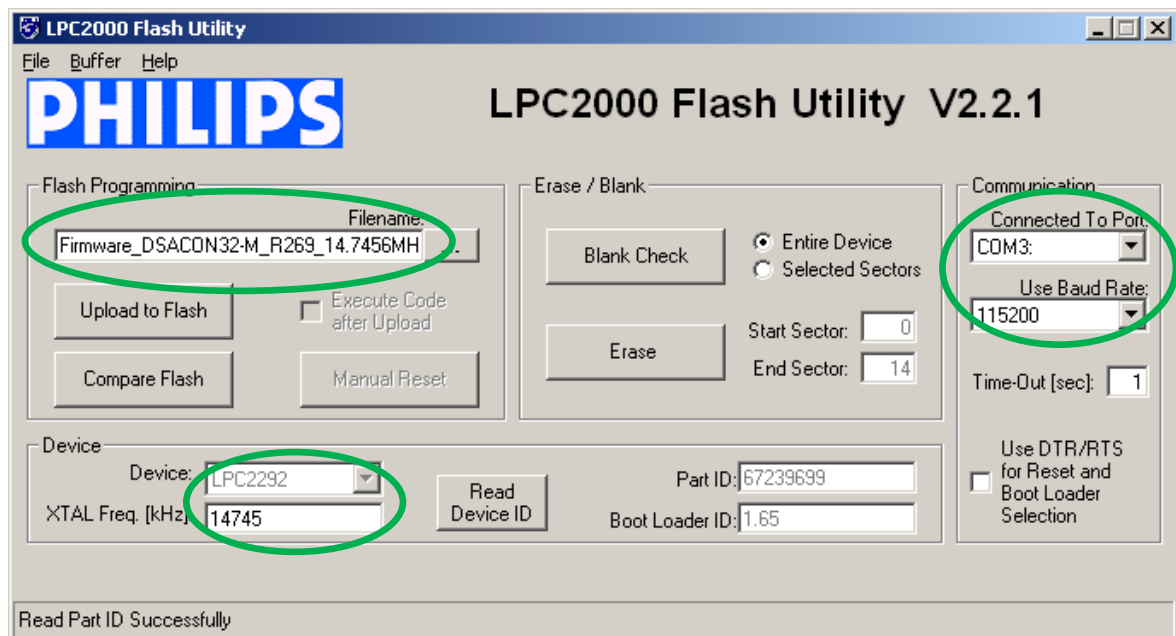
**Figure 5:** Response from an DSA32m controller with Device **LPC2294**, 12000 kHz.

- Step 9:** Depending on the hardware release found:  
 For DSA32m with LPC2292 go on with Step 10:  
 For DSA32m with LPC2294 go on with Step 13:

**Step 10:** For DSA32m with **LPC2292 14745 kHz**:

Adjust the “Filename”, “Use Baud Rate” and “XTAL Freq. (kHz)” as follows: (see also Figure 6):

- Flash Programming:
  - “Filename”: Select the “hex” file with the new DSA32m firmware provided by SCHUNK/Weiss Robotics, e.g. “Firmware\_DSA32-M\_R269\_14.7456MHz.hex”
- Communication:
  - “Connected To Port”: Select the RS232 COM interface which is connected to the RS232 channel 0 of the SDH. E.g.: **COM3**
  - “Use Baud rate”: **115200**
  - “Time-Out [sec]”: **1**
  - Deselect the “USE DTR/RTS” button
- Device:
  - “Device”: **LPC2292**
  - “XTAL Freq. (kHz)”: **14745**



**Figure 6:** Tool settings for updating DSA32m with 14745 kHz quartz

**Step 11:** Finally you can press the “Upload to Flash” button to actually perform the update.

This will take some time and a progress bar will appear. Please do not interrupt the process or power down the PC or SDH while the update is running.

When the process is finished the status line in the bottom of the flash tool window will state something like “File successfully uploaded”.

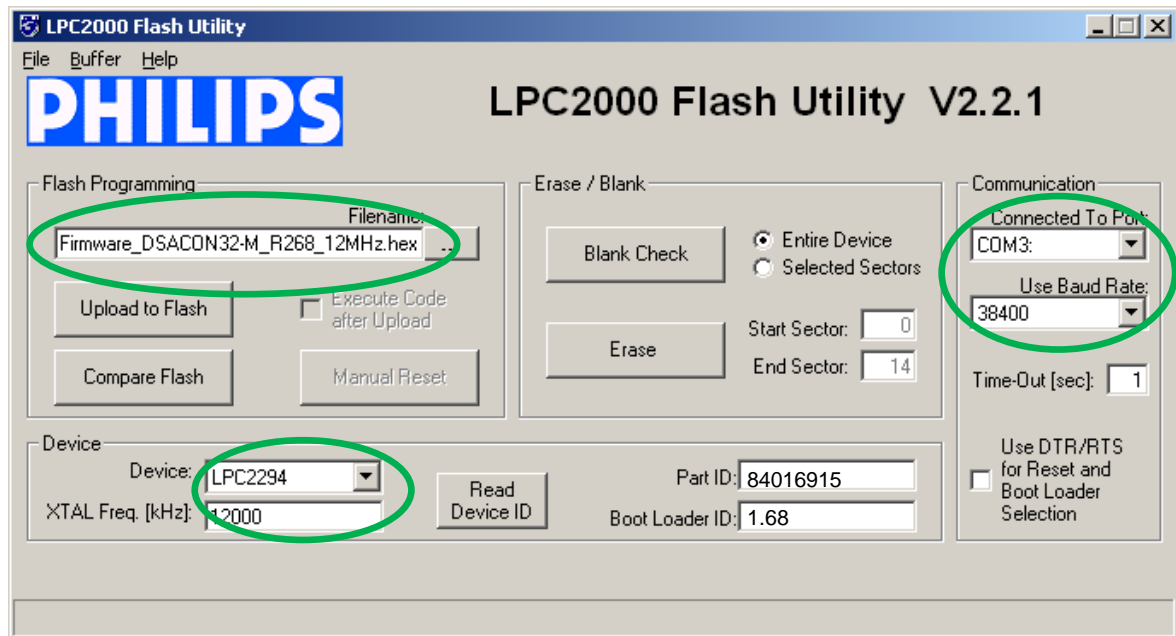
**Step 12:** In order to put the SDH back in normal operation mode, please reset the SDH or do a power down + power up cycle before using the SDH again.

**Step 13:** For DSA32m with **LPC2294 12000 kHz**:

Adjust the “Filename”, “Use Baud Rate” and “XTAL Freq. (kHz)” as follows: (see also Figure 6):

- Flash Programming:
  - “Filename”: Select the “hex” file with the new DSA32m firmware provided by SCHUNK/Weiss Robotics, e.g. “Firmware\_DSA32-M\_R268\_12MHz.hex”

- Communication:
  - “Connected To Port”: Select the RS232 COM interface which is connected to the RS232 channel **0** of the SDH. E.g.: **COM3**
  - “Use Baud rate”: **38400**
  - “Time-Out [sec]”: **1**
  - Deselect the “USE DTR/RTS” button
- Device:
  - “Device”: **LPC2294**
  - “XTAL Freq. (kHz)”: **12000**



**Figure 7:** Tool settings for updating DSACON32m with **LPC2294** and **12000** kHz quartz

**Step 14:** Finally you can press the “Upload to Flash” button to actually perform the update.

This will take some time and a progress bar will appear. Please do not interrupt the process or power down the PC or SDH while the update is running.

When the process is finished the status line in the bottom of the flash tool window will state something like “File successfully uploaded”.

**Step 15:** In order to put the SDH back in normal operation mode, please reset the SDH or do a power down + power up cycle before using the SDH again.

**Step 16:** In case you updated from a rather old firmware version then you might have to reconfigure the tactile sensor controller after updating. If after updating you cannot read tactile sensor info (neither with DSA-Explorer from Weiss Robotics nor with the demo programs provided by SCHUNK (demo-dsa, demo-tactile.py, ...)) then the following steps might be required:

- I. Switch SDH2 off
- II. Start a RS232 terminal program (like windows hyperterminal, PuTTY or Linux minicom, miniterm.py, ...) on the RS232 port where the tactile sensors are connected. Communication parameters: 115200 bit/s, 8 data bytes, no parity, 1 stop bit (8N1).
- III. Switch SDH2 on
- IV. The presence of the terminal program is detected by the DSACON32m and it will start an interactive ASCII based terminal session named “DSACON setup utility”
- V. Press Return once and ignore error messages, if any. Then enter the command “supervisor” and press Return.
- VI. Enter the password “kwfm78” plus Return
- VII. Enter the command “init sdh” plus Return





- VIII. Press “Y” plus return for Yes, if queried. This might be required several times.
- IX. Ignore errors like “OW-Bus fault”, if any
- X. Close the terminal program
- XI. Restart the SDH (power cycle)
- XII. The tactile sensors should now be readable again.

### **Troubleshooting**

In case that no proper connection can be established then please verify if you have used the proper settings for your actual DSACON32m hardware revision. If that is the case please read try over again. If you still cannot perform a proper update after some retries please double check the communication settings and connections.

If anything else fails please contact SCHUNK for further help.