Internal Software Tools Documentation

Documentation for internal software tools and customer development use.

1. libshim.dll / libshim.so

Mandatory shared library for interfacing with Jupiter amplifiers.

1.1. Recommendations

- Use the Python wrapper when possible, since more features are provided (e.g. unit conversions, reconnecting, etc.).
- If connection to the amplifier other Ethernet is dropped, connection can try be reestablished using `shim_soft_close` and then calling `ShimStart`.

1.2. Functional Descriptions

int shim_num_channels ()

Brief: Returns the number of amplifier channels.
Returns: `[int]` : Number of amplifier channels (rounds up to the nearest multiple of 8).

void shim_soft_close ()

Brief: Close network connection to the device (without altering the state of the amplifier). Useful for reestablishing the network connection.

int ShimStart (char* ifname, int name)

Brief: Start the connection to the Jupiter amplifiers.
Inputs:

- `ifname [char *]` : Network interface name
- `mode [int]` : 0 ~ Fast mode (immediate update); 1 ~ Safe mode (ramp update)
  Returns:
- `0 [int]` : Successful
- `-1 [int]` : Could not start device
- `-2 [int]` : Interface not valid
- `1 [int]` : Could not initialise device
- `2 [int]` : Could not initialise device configuration
- `3 [int]` : Invalid vendor or product ID
- `4 [int]` : Could not configure device mapping
- `5 [int]` : Could not reach Operational mode

void ShimStop()

Brief: Stop the connection to the Jupiter amplifiers.

int ShimEnable()

Brief: Same as `ShimEnableWithRamp` with `ramptime_ms=0`.
See also: ShimEnableWithRamp

int ShimEnableWithRamp (float ramptime_ms)

Brief: Enable the amplifier (turn on the internal feedback controller).
Description: Clears all setpoint values to zero and enable the internal feedback controller.
Pre-requisite: `ShimStart` must be called successfully before using this function.
Inputs:

- `ramptime_ms [float]` : Desired settling time for the controller (in milliseconds). This value will be truncated to the nearest value of the following: `0, 0.25, 0.5, 1, 2, 4, 8, 16, 32`. Use `0` as the default which will try to settle as fast as possible. NOTE: longer settling time will have increased robustness against disturbances.

int ShimDisable ()

Brief: Disable the amplifier (turn off the internal feedback controller).
Pre-requisite: `ShimStart` must be called successfully before using this function. If `ShimEnable` has not called, then this function will do nothing.
Description: All non-zero channels will be set to zero. Internal feedback controller will be turned off.
See also: ShimEnableWithRamp

void ShimResetCurr ()

Brief: Set all setpoint (i.e. shim current values) to zero.
Pre-requisite: `ShimStart` must be called successfully before using this function. If `ShimEnable` has not called, then this function will do nothing.
See also: ShimSetCurr

void ShimSetCurr(void* vals, int num, bool isfloat)

Brief: Set the shim current values.
Pre-requisite: `ShimStart` must be called successfully before using this function. If `ShimEnable` has not called, then this function will do nothing.
Inputs:

- `vals [void *]` : Pointer to array of integers or floats containing the shim currents that should be applied.
- `num [int]` : Number of channels (i.e. the length of the array `vals`). If the number of values is more than the actual number of channels, then the extra values are ignore. If the number of values is less than actual number of channels, then the value is not changed from the previous value that was applied.
- `isfloat [bool]` : If `false` then `vals` is cast as an array of integers and the values should be in *milliAmps*. Otherwise, if `true` then `vals` is cast as an array of floats and the values should be in *Amps*.
  See also: ShimResetCurr

int16_t* ShimGetAttr (int index)

Brief: Retrieve an attribute of the shim amplifiers.
Pre-requisite: `ShimStart` must be called successfully before using this function.
Inputs:

- `index [int]` : Index of the attribute to retrieve. 0 ~ Measured current (in mA); 1 ~ Status code (hex); 6 ~ Internal amplifier temperature (a.u.).
  Returns:
- `[int16 *]` Pointer to an array of attribute values. The length of the array is the number of detected physical channels; this can be retrieved using the `shim_num_channels()` function.

int ShimSetAttr (int* vals, int num, int index)

Brief: Set an attribute of the shim amplifiers.
Pre-requisite: `ShimStart` must be called successfully before using this function.
NOTE: This should be called *before* calling `ShimEnable` or `ShimEnableWithRamp` to be effective.
Inputs:

- `vals [int *]` : Array of attributes values to be set for each amplifier channel.

- `num [int]` : Number of channels (i.e. the length of the array `vals`). If the number of values is more than the actual number of channels, then the extra values are ignore. If the number of values is less than actual number of channels, then the value is not changed from the previous value that was applied.
- `index [int]` : Index of the attribute to modify. 15 ~ Load resistance (in milliOhm); 16 ~ Load inductance (in microHenry).
  Returns:
- `[int]` : Number of channels that had their attributes set.

int ShimChannelDiverged ()

Brief: Check if all shim channels have a similar value to their respective setpoint currents. Threshold is +/-50mA.
Returns:

- `0 [int]` : All channels have converged to their setpoint currents.
- `>=1 [int]` : The channel number of the first channel that where the measured current is not similar to the applied setpoint current.

---

2. hwio.pyd / hwio.so

Python shared library that wraps the libshim.dll / libshim.so library.

2.1. Recommendations

- Use `LoadShimSets` instead of `ApplyShimSet` whenever possible

2.2. Functional Descriptions

LibVersion() -> str

Brief: Return the version of this Python module (as string).

DriverVersion() -> str

Brief: Return the version of the C-shared library libshim.dll / libshim.so (as string).

JupiterDeviceInfo() -> str

Brief: Returns network interface names and descriptions. Returns "" if an error occurred.

LinkupHardware (ifname: string, mode: int, opts: dict)

Brief: Connect to hardware devices.
Description: Continually retries to connect, if a connection could not be established. If a
hardware input could not be detected, then no hardware input is used. If a connection to the same device current exists, then the connection will first be closed before linking up to the hardware again.
Inputs:

- `ifname [string]` : Name of the network interface device. See `JupiterDeviceInfo()`
- `mode [int]` : 0 ~ Fast-mode (does not ramp to the shim values but applies the currents immediately); 1 ~ Safe-mode (ramps up to the applied shim values at a rate of 1A / sec updated at 20ms intervals).
- `opts [dict], optional:` : Options for specifying the hardware input. If this is empty or None, then no input is used. The input can be used for trigger-based or real-time shimming. `opts[pin]` ~ The pin number of the device. For Windows, this values is between 1 and 3 (typically 3); `opts[mode]` ~ 0=Rising-edge triggered, 1=Falling-edge triggered, 2=Analog mode. If `opts['mode']` is 0 or 1, then every time an event is triggered the next set of shim values is applied to the amplifiers (see `LoadShimSets`). If `opts['mode']` is 2, then the input is treated as an analog input and every time the input is sampled, the function `ShimRealtime` if called from the `shimplugin.dll\.so` shared library. `opts['ramptime']` ~ Settling time (in milliseconds) for the setpoint to be reached. Increasing this makes the amplifier more robust against disturbances. Note: This is not the same as using `mode=1` in this

function. When set to 0, the amplifier will try to update as fast as possible. Ramp-times are rounded to the nearest value of either 0.25, 0.5, 1, 2, 4, 8, 16, or 32.

CloseHardwareLink()

Brief: Stop threads and close all devices.

HardwareReport (full: bool=false) -> dict

Brief: Returns status attributes of each channel of the amplifiers.
Description: Returned attributes include attributes such as status codes, and internal temperatures. If no communication could be made with the amplifier then the only key in the returned dictionary is `connected`.
Return:

- `result['connected']`: 0 ~ No amplifier connected; 1 ~ Amplifier connected.
- `result['diverged']`: 0 ~ Measured currents match the applied setpoints; 1 ~ The measured current of at least one channel does not match the applied setpoint.
- `result['status']`: List of status codes (to be used for diagnosis).
- `result['curr']`: List of measured current values.
- `result['temp']`: List of internal temperatures (on-board the amplifier).

SetShimLoad (resistance: list[int], inductance: list[int])

Brief: Manually set the load characteristics for each channel. This must be called *before* linkup to the hardware (i.e. before calling `LinkupHardware`)
Inputs:

- `resistance`: List of resistances in milliOhm to which each load channel should be set.
- `inductance`: List of inductances in microHenry to which each load channel should be set.

LoadShimSets (filename: str, delay: int=0)

Brief: Upload a file containing shim values to the amplifiers.
Description: The first row of values will be applied when this function is called. To go to the next row of shim values, either use `ApplyShimManually` or use an external trigger as an input (only if `LinkupHardware` was called with appropriate settings, e.g. falling- or rising-edge mode).
Inputs:

- `filename [str]`: File path of the text file with the shim current values. The first row of values will be applied, then the next rows can be applied using a trigger (if the hardware input was set to either 'rising edge' or 'falling edge' mode, see `LinkupHardware`), or by calling the function `ApplyShimManually`.
- `delay [int], optional`: Duration to wait (in ms) before applying the next set of shim values. Default: 0.
  Raises: Exception when the text file cannot be parsed. Must be a space delimited text file of float values.
  NOTE: This will only be used when the hardware input is not in 'Analog' mode, see `LinkupHardware`. If the hardware input was configured to 'Analog' mode, then the 'shimplugin.dll / shimplugin.so' shared library is used.

ApplyShimManually (command: str)

Brief: Manually apply a predefined set of shim values.
Inputs:

- `command [str]`: `next` ~ Apply the next row of shim current values (see `LoadShimSets`); `prev` ~ Apply the previous row of shim current values (see `LoadShimSets`); `reset` ~ Reset all the shim values to zero.
  Raises: Exception when command could not be applied.

ApplyShimSet (shims: list[float]) -> int

Brief: Apply a set of shim values to the amplifiers.
Inputs:

- `shims [list[float]]` : Sets the amplifier current to the values in this list (units are in Amps). Any values outside of the range -4.0 to 4.0 A will be clipped.
  Returns:
- `int` : Analog value if an analog input is available. Usually this can be discarded.
  NOTE: Preferrably use `LoadShimSets` whenever possible.

---

3. Oolong

3.1. Summary

Oolong is a terminal tool for controlling the Jupiter amplifiers. It is intended as an alternative to Sinope to be used by MR Shim employees internally (not for distribution to customers).

3.2. Configuration

See the Sinope manual for instructions on how to use the configuration file. *Oolong* is only compatible with newer config file format (i.e. Sinope 3 format with sections).
The only difference is for `gpio=-1` where *Oolong* will simulate a trigger.

3.3. Usage

Instructions for usage are given in `oolong -h` which will output the following:

```
oolong - interface for controlling and monitoring Jupiter amplifiers
Usage:
    oolong [filename]
    oolong [options] [filename]

Options:
    -m:[val], --mode:[val]    Select mode to use ('safe', 'fast', 'update')
                              'safe' : Ramps to shim current values.
                              'fast' : Immediately apply current values.
                              'update' : Update firmware on amplifiers.
    -f, --fall    Falling edge-trigger for shim set updates.
    -h, --help    Display help.
    -i, --info    Display hardware device information.
    -k, --key     Generate and display key (for generating a valid license).

Arguments:
    In 'update' mode (see `--mode`), the file is used to update the firmware.
    Otherwise, the file should be a text file with shim current values (in
    Amps). Each column is a channel of the amplifiers. Each row is a set that
    can be applied with an external trigger (default is rising-edge triggered,
    see also `--fall`).
    If no file is given, then the device is open in real-time mode, i.e.
    continually updating the shim current values based on the input analog
    signal and the shimplugin.so library.

Copyright (C) MR Shim GmbH - All Rights Reserved.
E-mail info@mrshim.de for more information.
```