Chapter 1

Solstis class

One or two Solstis objects can be instantiated. Once a solstis is initialized, the tcpip connection between computer and solstis ICE-box should be established. After this is successfully done, all other functions can be used. Most functions have as output a struct *response*, that contains info from the solstis laser (whether the command was successfully executed, and more). Some functions use the response of other functions, and response can be used for debug purposes. For most functions, this output can be suppressed.

MATLAB EXAMPLE

handle = **Solstis.getInstance(** *'192.168.1.221'* , *39933* **);**
response = *handle.***OpenTCPIP();**
response = *handle.*[anyFunction];

Properties

- *last_WL*
  The last wavelength as checked with function GetWL() or set with GoToWL().

- *sol_ip_address*
  Ip-address of the solstis ICE-box.

- *pc_ip_address*
  Ip-address of the computer controlling the solstis.

- *port*
  TCPIP port number of solstis.
- *write_reports*
  Boolean that determines whether all function responses are saved as reports in *report*. Default value is false.
- *report*
  Cell with all report since initialization of solstis. Can be used for debug purposes, by setting *write_reports* to true.
- *tcp*
  tcpip object with connection to the solstis.
- *terascan*
  Struct with terascan settings. Set with function TeraScanInit().
- *channel*
  Number (1,2,3 or 4) of this solstis at the wavemeter switchbox.
- *lambda_lock*
  String that shows whether lambda_lock is on.

Functions

handle = **getInstance(** *ip_address* , *port* **)**

Instantiates a solstis object and puts it under the name *handle* (or any other name that is put before the = sign). *ip_address* is a string with the ip-address of the ICE-box of the solstis on the local network connected to the lab pc. *port* is an integer with the port number. This can be found in the soltis settings if necessary.

response = **OpenTCPIP()**

Opens the tcpip connection with the solstis ICE-box.

response = **Query(** *query* **)**

Used by other functions to send commands to the solstis over tcpip, and wait for its response. Query() uses the function Parse() to change the solstis response from a string to a struct with fields for each piece of info contained in the string. *query* is a string, formatted as described in the 'Solstis_3_TCP_JSON_protocol_v19.pdf' documentation, given by M-Squared.

response = **Parse(** *ret* **)**

Used by other functions to change a return string *ret* from the solstis into a *response* struct that contains each piece of info as a separate field.

response = **WaitForResponse()**

Used for solstis commands that give a so-called 'final_report' some time after the command is issued. This blocks matlab execution until a response is obtained. Used in function GoToWL().

[response , wavelength] = **GetWL()**

Reads the current wavelength, as seen by the wavemeter, and outputs it in *wavelength*. Also *response.status* is used to check whether the solstis has a good connection with the wavemeter (used for terascans). However, it is recommended to measure the wavelength directly from the wavemeter, since this is much faster than via the solstis.

response = **GoToWL(** *wavelength* **)**

Issues the solstis to go to *wavelength* in nanometers and lock the etalon there. The resolution of this can be set in the solstis settings (minimum 0.00001 nm). Once the requested wavelength is reached, no further locking to this wavelength is performed, unless lambda lock is on. This function blocks execution until the 'final_report' is received from the solstis. GoToWL() could take between seconds and minutes to finish.

response = **Lock(** *onOffStr* **)**

Turn on or off the lambda lock function. *onOffStr* is a string that should either be *on* or *off*. When lambda lock is turned on, active feedback keeps the solstis at the wavelength it is on at that precise moment. When using GoToWL(), the new wavelength is saved as the wavelength to lock at. N.B.: scanning the solstis with either FastScan() or TeraScan() is not possible while lambda lock in on.

bufferRemains = **ClearBuffer()**

Checks if there are any bytes in the output message buffer of the solstis and reads them into *bufferRemains*. Used for checking the status of Terascans and at the end of scripts to prevent bugs.

response = **TeraScanInit(** *scan_type , start , stop , scan_rate , units* **)**

Initialize terascan settings. String *scan_type* is either *'medium'* or *'fine'*. Number *start* is the start wavelength of the scan in nanometers. Number *stop* is the stop wavelength of the scan in nanometers. Number *scan_rate* combined with string *units* gives the approximate scan speed. *scan_rate* is an integer, *units* is either *'MHz/s'* (only for fine scan) or *'GHz/s'* (both fine and medium). Available scan rates are different for medium and fine scan: Allowed for fine are $[1, 2, 5, 10, 20, 50, 100, 200, 500]$ MHz/s and $[1, 2, 5, 10, 20]$ GHz/s. Allowed for medium are $[1, 2, 5, 10, 15, 20, 50, 100]$ GHz/s.

response = **TeraScan(** *finalreport* **)**

Start a terascan. When a terascan was previously configured, this command will start that scan. Otherwise, the terascan should first be configured using TeraScanInit(). String *finalreport* should be 'on' or 'off'. When on, this will generate a report in the solstis message buffer when the terascan has finished.

response = **TeraScanOutput(** *operation , pause* **)**

Used to get automated responses during terascans. When *operation* is set to 'start', the solstis will write an update report string to its buffer at the start and end of every scan segment. 'stop' will prevent automated feedback.

pause is either 'on' or 'off', and lets the solstis pause at the start of every scan segment, waiting for the function TeraScanContinue(). For performing scans with a single solstis using the Solstis object functions, it is most convenient to set *operation* to 'stop'.

response = **TeraScanContinue()**

Use to continue a paused terascan. See TeraScanOutput().

response = **TeraScanStatus()**

Receive a report with the status of the current terascan. Not used currently.

response = **FastScan(** *width , duration* **)**

Used for short range scans. Currently, only a single resonator scan is supported, with a range of max $\approx$ 30 GHz. *width* is either a single numeric value that depicts the scan width in GHz, centered around the current wavelength, or a two value numeric array that depicts the start and stop wavelength of the scan in nanometers. *duration* is the time the whole scan will take in seconds (min 0.01, max 10000), and effectively determines the scan speed and the number of measurement points per GHz scan range. If a higher number of measurement points per GHz is required, increase the duration or reduce the scan width. N.B.: The actual scan width can differ up to 20% from the requested width, due to imprecise resonator calibration.

response = **PollFastScan()**

Check the status of the current resonator fastscan. *response.status* is used to check if a fastscan is finished.