OpenTCI

Open Telescope Control Interface

An open specification of a TPL2 based interface to control a telescope

Version 2.5

Document Id:

4PI-DOC-03-008-02-3

Michael Ruder, Daniel Plasa

Revision 369 as of March 3, 2008 (michael)

Revision History:

10.09.2003	Initial Version (dp)
28.06.2005	Finalized 2.2 (dp)
28.02.2006	2.3 Updates for non AZ-ZD mount types (dp)
27.07.2006	2.4 Updates on dome access (dp)
17.01.2007	2.5 Updates on dome access (II) (mr)

$Id: OpenTCI.tex 369 2008-03-03 12:34:11Z michael$

Contents

1 Preface 1

2 Basics 1

3 List of Basic Modules 1
3.1 Testing the availability of a module . . . 2
3.2 Common module variables . . . 2

4 Errors, Logs and Events 4

5 Axis Modules 5
5.1 A Generic Axis Module . . . 5
5.1.1 Module TRAJECTORY . . . 9
5.2 Modules AZ, ZD[] and DEROTATOR[] . . . 10
5.3 Module FOCUS . . . 10
5.4 Module FILTER[] . . . 10

6 Module CABINET 11
6.1 Hardware Independent Functions . . . 11
6.2 Almost Independent Functions . . . 13
6.3 Hardware Dependent Functions . . . 14

7 Module CONFIG 14

8 Module LOCAL 15

9 Module COVER[] 16

10 Module DOME[] 17

11 Module GPS 17

12 Module MIRROR 18

13 Modules SENSOR[] and SWITCH[] 18

14 Module POINTING (optional) 18
14.1 Variables of TARGET and CURRENT . . . 21
14.2 Pointing model parameters . . . 21
14.3 Refraction parameters . . . 22

A Quick guide for the impatient **23**

B TPL2 Error and Event numbers **24**

C Coordinate Systems of the Telescope Axes **25**

C.1 Azimuth, Zenith Distance and Focus 25

C.2 Derotators, Filters and M3 rotation 25

C.3 Covers and Domes 25

C.4 Other coordinates 27

D Abbreviations **27**

References **27**

1 Preface

This document describes the telescope control interface, an open network interface to a telescope control software (TCS). For the communication the TPL2 protocol shall be used (refer to [1] for protocol specifications). This interface is designed to be as hardware independent as possible but there are of course several functions that are different or even unavailable on some telescope hardware.

2 Basics

- The OpenTCI defines a list of modules which will give an ordered, hierarchical (almost) hardware independent access to all important telescope functions.
- Depending on the telescope hardware, only a subset of the modules listed below must be implemented.
- A generic error and log handling interface is specified.
- For debugging and problem analysis there may also be modules which give access to internal settings, like controller settings, cabinet switches etc. These structures will be hardware dependent and may differ in various implementations. However any implementation should aim — by design — to make normal telescope operation possible without accessing these modules.
- In addition to these basic modules there may be additional modules covering individual needs and functions. These modules shall be described in separate documents.

3 List of Basic Modules

The following modules are specified by the current version of the OpenTCI. The minimal subset of modules is marked with an **R**:

Name	Function	
R^+	`AZ`*	Control of the telescope azimuth
R^+	`ZD[]`*	Control of the telescope zenith distance
R^+	`HA`*	Control of the telescope hour angle
R^+	`DEC`*	Control of the telescope declination
R	`CABINET`	Everything that is related to the control cabinet
R	`CONFIG`	Configuration parameters and information (like the telescope mount type etc.)

Name	Function	
	COVER[]	Covers that can be controlled (e.g. mirror covers)
	DEROTATOR[]	Control over the image derotators
	DOME[]	Control over the dome
	FILTER[]	Control over the filter wheels
	FOCUS	Control over the focus motor
	GPS	Access to the GPS receiver for time and location data
R	LOCAL	UTC, UT1, TAI, geographic position of the Telescope
	MIRROR	Mirror control (control of a flip mirror)
	SENSOR[]	Any sensors (temperatures, pressure)
	SWITCH[]	Any switches (pumps etc.)
	POINTING	Module for direct astronomical functions

$^{+}$) At least two of them are needed, most commonly being (AZ/ZD) or (RA/DEC).

*) Modules availability may depend on the telescope hardware / possible mount options. E.g. ZD[] is a plain module for AZ-ZD mount telescopes (ZD) and an array of size 2 for ZD-ZD mounts (ZD[2]).

3.1 Testing the availability of a module

The user can rely on the existence of the required modules. The availability of modules that are not required can be tested by reading the VERSION variable which must return a non-zero value if the module exists[1]. If a module is not available, it may also be missing at all, resulting in an OpenTPL error message, reporting this module as unknown.

3.2 Common module variables

Every module must have at least the following variables, which will not be listed in the following detailed module description sections unless their meaning is changed:

Name	Type	Access	Description
VERSION	INT	RO	Interface version (see below)
TYPE	INT	RO	Hardware type (≥ 1, see below).

[1] until OpenTCI 2.3 the TYPE variable was also required to have a non-zero value to indicate the availability of the module. However since this information is redundant, as of version 2.5 only VERSION is required to be non-zero.

Name	Type	Access	Description
ERROR_STATE	INT	RW	If this variable is $\neq$ 0, then the axis is currently having errors possibly preventing proper operation (see below) Writing 0 to it will acknowledge the error condition and try to resume normal operation.

Every module must contain a VERSION variable. It shall give the client information about the structure of the interface layout. Reading it will return an interface version, an interface age and a revision number (packed into one number):

- The interface version will be increase whenever variables are added or removed. The latest OpenTCI version is **24**.
- The interface age allows clients to check if the expected interface version can be used even if the interface version is different from the expected version. The age is computed as maximum supported interface version minus minimum supported version.
  Example: interface version is 3 but all variables of version 2 are still in place to stay compatible with old clients. Therefor the age equals $3 - 2 = 1$.
- The interface revision indicates the code revision. In general: the higher, the better, fewer bugs etc.

The three numbers must be packed into a single integer with hexadecimal representation 0x---IIAARR, where --- is unspecified, II is the interface version, AA is the age and RR the revision number.

The module must also have a TYPE variable, which shall give the client specific information about the type of featured hardware controlled by the module[2].
The module must also have a ERROR_STATE variable, which shall give the client information about the status of the hardware controlled by the module. See 4 for more details.
Some of the modules are defined as an array (shown as []) to allow multiple instances. If only one instance is available, the module should not be defined as an array.
On reading variables, unless other noted the value must be returned immediately, i.e. within some 5 ms. On writing variables the behavior is described with every writable variable.

[2]since this variable was also used to indicate the availability of the module up to OpenTCI 2.3, the types uded should always be ≥ 1

4 Errors, Logs and Events

Errors can arise from several different sources. One big source of errors is of course the telescope hardware itself (motor failures, broken fuses etc.). Other errors may arise during normal operation, e.g. one user sets an axis target position and waits for the axis to reach that position while another user resets that position to another value and user one's position will never be reached. The first class of errors is normally strictly hardware dependent, the latter class is independent. Therefore the OpenTCI defines two ways of error notification.

- Every module representing a functional group of the telescope has a `ERROR_STATE` variable. It will be either 0 if the hardware is fully working or contain binary coded the state of useability:

Bit	Description
0	PANIC: module does not work anymore; the error is so severe that the whole telescope cannot operate anymore.
1	ERROR: module has errors and is not working.
2	WARNING: module is still working but performance may be influenced.
3	INFO: modules is working; some information on the status have been saved.
4	DEBUG: modules is working; some debug information is available.
5+	reserved: must be 0.

  TPL2 events can be utilized to notify the client that errors have occurred. Writing 0 to the variable will acknowledge the error condition and allow the controlled hardware to resume normal operation, unless of course the error causing condition has not been yet resolved.

- The (`CABINET.ERROR_STATE` variable will return all module `ERROR_STATES` combined with a logical *OR*. In addition, errors that cannot be assigned to a specific functional group (mainly in supporting hardware, e.g. UPS, fans etc.) will be reported here. This variable can be used to determine the state of the telescope at a glance.

- A special module (`CABINET.STATUS`, see 6.2) will provide structures to read out the hardware dependent error memory. It will also provide methods to clear the error memory.

- A special module array (`CABINET.STATUS.LOG[]`) shall give an hardware independent way of reading log files from the TCS.
- Errors that occur during telescope operation by user interaction or by wrong user input (as described in the example) shall be notified by TPL2 event messages. They must be returned as TPL2 error code by the called OpenTCI variable.
- In addition, implementations of an OpenTCI telescope control should provide TPL2 events to inform the client about important state changes, e.g. axis is now referenced, trajectory is now running etc. OpenTCI defines a list of error / event numbers, see appendix B.

5 Axis Modules

5.1 A Generic Axis Module

Since many basic modules describe axes that provide more or less the same interface, a generic axis module will be described here. The description of the individual modules will refer to this section. A generic axis provides the following functions:

Name	Type	Access	Description
`POWER`	INT	RW	Switch power for this component. Possible values are: 0 = off, 1 = on. The main power (`CABINET.POWER`) has to be turned on to enable or disable power for single components, otherwise this command will fail. Set will only return on set if desired value is reached or on error.
`POWER_STATE`	FLOAT	RO	Power state of the component. Possible values are: -1.0 = emergency off, 0.0 = off, 0.75 = standby, 1.0 = on.
`ERROR_STATE`	INT	RO	If this variable is $\neq$ 0, then the axis is currently having errors possibly preventing proper operation (see above).

Name	Type	Access	Description
LIMIT_STATE	INT	RO	If this variable is $\neq 0$, then the axis is currently at some limit. Bit0: Hardware limit minimum position. Bit1: Hardware limit maximum position. Bit7: Hardware limit is blocking movement. Bit8: Software limit minimum position. Bit9: Software limit maximum position. Bit15: Software limit is blocking movement.
MOTION_STATE	INT	RO	If this variable is $\neq 0$, then the axis is currently moving. Bit0: Axis moving. Bit1: Trajectory is being executed. Bit2: Something is blocking movement (e.g. limit switches). Bit3: Axis has acquired current axis target position or position deviation during trajectory execution is below threshold. Bit4: Axis movement is limited by max. speed / acceleration or jerk. This Bit indicates that e.g. trajectory points are to quickly changing axis target position.
REFERENCED	FLOAT	RO	Referenced state of this component. Possible values are: 0.0 = not referenced, 1.0 = referenced, 0.0 . . . 1.0 values will indicate the progress during referencing.
REINIT	INT	WO	Redo initialization of this component. Possible values are: 0 = initialize only parts that are not already initialized, 1 = discard current state and force reinitialization. Will return immediately.

Name	Type	Access	Description
REALPOS	FLOAT	RO	True current position in component specific units. For the position of the reference value 0.0 see appendix C for an explanation of the coordinate systems. With the properties !MIN and !MAX the accessible range of positions can be queried. If either of the properties is NULL this will indicate that there is no limit for the position in that direction. E.g. the azimuth of a telescope might rotate infinitely.
CYCLIC	FLOAT	RO	In the above described case of unlimited rotary movement, the periodicity of movement can be queried here. Mots likely this will yield 360.0.
CURRPOS	FLOAT	RO	The current position, possibly corrected by the given OFFSET (in the same unit as REALPOS).
CURRSPEED	FLOAT	RW	The current speed of the axis in units/s. By writing, the maximum speed can be reduced to a number $\leq$CURRSPEED!MAX[3]; will return immediately.
CURRACC	FLOAT	RO	The current acceleration in units/s^2.
CURRJRK	FLOAT	RO	The current jerk in units/s^3.

[3]Note, that OpenTPL currently will unfortunately not reflect this by changing the MAX property since properties are considered static.

Name	Type	Access	Description
TARGETPOS	FLOAT	RW	The position the axis is currently moving to (in the same unit as REALPOS), possibly corrected by the current OFFSET. While a trajectory is executed, the variable will constantly be updated to the current position on the trajectory. By writing to this variable, the axis will switch into direct positioning mode, cancelling a possibly running trajectory. On Set, the command will return after the position has been reached (i.e. MOTION_STATE:3 is set), the target was reset by another request or an error occurred, so the execution of this command may take a while. Meanwhile it is possible to set another target position. However this will terminate previous TARGETPOS commands.
OFFSET	FLOAT	RW	Additional offset (in the same unit as REALPOS) that will be added to *all* positioning requests (both direct positioning and trajectories) for that component. This can be used for a guiding system. It will also be used to implement the manual movement with an optional paddle if CABINET.MANUAL_CONTROL.ACTIVE is set. On Set, the command will only return after the component has reached the given offset (i.e. MOTION_STATE:3 is set), another offset was requested or an error occurred Meanwhile it is possible to set a different offset. Again, this will terminate previous offset commands.
TARGETDISTANCE	FLOAT	RO	The distance between CURRPOS and TARGETPOS, also valid during trajectory execution.

Name	Type	Access	Description
STOP	INT	WO	Stop axis movement immediately with maximum possible jerk / deceleration (may be higher than the values in CURRACC!MAX and CURRJRK!MAX). Will return immediately.
TRAJECTORY	MODULE	—	Trajectory handling for the axis (optional).

5.1.1 Module TRAJECTORY

This module allows movement of an axis following a given time / position path — a trajectory. It can be used to implement highly precise tracking.
Any given OFFSET will be applied to the trajectory points. It is possible to change the offsets during trajectory execution, allowing e.g. to center a star during tracking.
The following functions are provided in this module:

Name	Type	Access	Description
BUFFER[]	STRUCT		Struct array which is used as buffer for new sample points (see below).
ADDPOINTS	INT	RW	On write: will add the number of buffer points (starting from BUFFER[0]) into the internal trajectory handling. Will return immediately. On read: returns the number of written points that have been written on the last write access.
FREEPOINTS	INT	RO	This is the number of buffer points that can be added without blocking.
RUN	INT	RW	By writing 1 to this variable, the execution is started. This function will only return if the internal trajectory buffer is empty or the execution was aborted. During the trajectory execution, more points can be added to allow continuous movements over longer periods of time. By setting the variable to 0, the execution is aborted and all internal buffers are freed. To check if a trajectory is executing, the variable can also be read, returning 0 or 1.
STARTTIME	FLOAT	RO	Time of the first trajectory point.

Name	Type	Access	Description
RUNTIME	FLOAT	RO	Remaining runtime of the trajectory.

The struct array for the trajectory buffer has the following entries:

Name	Type	Access	Description
TIME	FLOAT	RW	Time in UT1 (Seconds since 01.01.1970 with fractions after the decimal point).
TARGETPOS	FLOAT	RW	Position in the same unit as TARGETPOS.

5.2 Modules AZ, ZD[] and DEROTATOR[]

These modules are generic axes as described above. All units are given in degrees. They provide direct control of both telescope main axes and the telescope's derotator(s). For tracking of astronomical objects, the respective TRAJECTORY sub-modules should be used which shall allow time synchronized movements.

Note: whether an AZ-ZD mounted telescope is equipped with one or more derotators can be determined by checking the DEROTATOR[].TYPE variable. The index of the DEROTATOR[] determines the mount position of the derotator on the telescope (e.g. Nasmyth 1 / 2 etc.).

Note: if an ZD-ZD mounted telescope is equipped with a derotator (i.e. an azimuth rotator) this derotator will still be accessible through the DEROTATOR[0] module instead of a AZ module.

5.3 Module FOCUS

This module provides functions to change the focus position (usually by moving the M2 mirror) and is a generic axis with millimeter unit. **Note**: this axis does not support timed movements and therefore has no TRAJECTORY sub-module.

5.4 Module FILTER[]

This module array provides functions to move filter(s) that are mounted to special positions at the telescope. It is a general axis without a TRAJECTORY sub-module, the index is defined according to the DEROTATOR index and the unit is unspecified. **Note**: Depending on the filter device, CURRSPEED and CURRACC might not provide valid data or may not be used at all. Also depending on the kind of filter device (discrete or continuous) the *POS variables might only be set to integer values.

6 Module CABINET

This module allows direct access to the telescope's control cabinet and provides functions for turning on and off main power as well as a lot of diagnostic functions. Since every telescope will have a control cabinet (or equivalent device), the `TYPE` variable will never be zero. In fact, it should be a number that can identify the manufacturer of the cabinet and the structure of the debug functionality unambiguously. This number is meant for internal use only and the client should not rely it. This module will contain both hardware dependent and independent functions.

6.1 Hardware Independent Functions

The following hardware independent functions shall be provided:

Name	Type	Access	Description
`POWER`	INT	RW	Switch main power on or off. Possible values: 0 = off, 1 = on. Individual components can only be turned on and off with their respective `POWER` variable, if the main power has been turned on. After turning main power on or off, all components will be powered on or off accordingly. Set will only return on set if desired value is reached or on error.
`POWER_STATE`	FLOAT	RO	Main power state. Possible values are: −1.0 = emergency off, 0.0 = off, 1.0 = on. The power state of individual components can be checked with the respective `POWER_STATE` variable of the component.
`REFERENCED`	FLOAT	RO	Referencing state of all components. Possible value are: 0.0 = no component referenced, 1.0 = all components referenced, 0.0 . . . 1.0 values will indicate the progress during referencing.

Name	Type	Access	Description
			The referencing state of individual components can be checked with the respective REFERENCED variable of the component.
REINIT	INT	WO	Redo initialization of all components.Possible values are: 0 = initialize only components that are not already initialized, 1 = discard current initialization and force re-initialization. Will return immediately.
MODE	INT	RO	Operation mode of the telescope. Possible values are: −1 = only local operation permitted, 0 = no operation permitted, 1 = remote operation permitted (i.e. via the OpenTCI interface).
MANUAL_CONTROL	MODULE	—	Information about the optional bottle for manual control (see below).
SETUP	STRUCT	—	Access to detailed cabinet information (see below).
STATUS	MODULE	—	Access to status/error flags (see below).
STOP	INT	WO	Stop all movements immediately. Will return immediately.

The CABINET.MANUAL_CONTROL module gives access to information and settings of the optional manual control bottle.

Name	Type	Access	Description
ACTIVE	INT	RW	Manual control activation state. Possible values: 0 = disabled, 1 = enabled. Set will only return on set if desired value is reached or on error.
BRIGHTNESS	FLOAT	RW	Brightness of buttons. Possible values range from: 0 . . . 1.0 (lowest . . . brightest setting). Set will return immediately.
MODE	INT	RO	Mode of the manual control (e.g. key switch on the paddle). Possible values are:

Name	Type	Access	Description
			0 = disabled, 1 = enabled. Note that the actual state of the manual control is an "AND" conjunction of `MODE` and `ACTIVE`.
`SELECTION`	INT	RO	A number representing a selection (e.g. selected device) that was made with the manual control.

The `CABINET.SETUP` structure gives access to information (such as name of the manufacturer, software version, configuration etc.) of the cabinet itself.

Name	Type	Access	Description
`MANUFACTURER`	STRING	RO	Name of the company that built the control cabinet.
`VERSION_TEXT`	STRING	RO	Concatenated string of all version identifiers of the used components.
`HW_ID`	STRING	RO	Unique hardware ID of the cabinet.
`HW_VERSION`	STRING	RO	Version of the hardware.
`SW_ID`	STRING	RO	Unique ID of the low level driver software running on the cabinet computer.
`SW_VERSION`	STRING	RO	Version of the low level driver software.
`CONFIG_ID`	STRING	RO	Unique hardware ID of the control components mounted on the telescope.
`CONFIG_VERSION`	STRING	RO	Version of configuration.
`PARAM_ID`	STRING	RO	Unique ID for the current set of parameters for the low level driver software.
`PARAM_VERSION`	STRING	RO	Version of the current parameter set.
`PARAM_NAME`	STRING	RO	Name of the current parameter set.

6.2 Almost Independent Functions

The `CABINET.STATUS` module allows the client to access the telescope system status and log files. Whenever the telescope system discovers something unusual during operation, it will save this event in a list. The list entries will be kept until they get cleared by the client.

Note: Even though the conditions that caused an events may have been resolved in the meantime, the list entry stays until cleared by the client.

The CABINET.STATUS module contains both hardware dependent and independent information. The events (see below) are hardware dependent. However the hardware dependent events are classified by by the bits described in section 4 to allow a hardware-independent assessment of the severity.

Name	Type	Access	Description
GLOBAL	INT	RO	All ERROR_STATES logically OR'd.
LOG[]	STRUCT		Access to the system log files.
LIST	STRING	RO	A comma (,) separated list of all saved error events. The format is: Age:Error:Status:TotalCount:Count:Active:Device[:optional comment],...
CLEAR	INT	WO	On writing −1, all events will be deleted.

Age will give the age of occurrence in seconds, Error is a symbolic error name, Status is a number containing the bit-field in the above described meaning, TotalCount how often the error was reported since starting the telescope, Count the counter since the last acknowledge, Device contains the source of the Event[4]. An optional comment field can be omitted.

6.3 Hardware Dependent Functions

In addition to the functions described above there may be several other structures and variables available, mostly for debugging and maintenance tasks. These are however strongly hardware dependent and cannot be described here as they are object to an actual implementation. These structures shall however be situated only in a CABINET.CORE sub-module.

7 Module CONFIG

This module shall be used to gather configuration information about the telescope in general.

[4]The device needs not necessarily be represented by an OpenTCI module, however it most often will be

Name	Type	Access	Description
MOUNTOPTIONS	STRING	RO	A string containing a comma separated list of useable axes combinations to achieve telescope positioning, e.g. AZ-ZD, RA-DEC or ZD-ZD.
MOUNT	STRING	RW	The currently active mount configuration. Same as given in MOUNTOPTIONS. If setting this variable the return behavior is undefined. It might even end your connection if the used TPL2 server does not support dynamic re-configuring.
ORIENTATION	FLOAT	RW	Tube orientation for ZD-ZD mounted telescopes when ZD[0].REALPOS=0 in geodetic coordinates. If setting this variable the return behavior is undefined. It might even end your connection if the used TPL2 server does not support dynamic re-configuring.
DOME.CAPABILITIES	INT	RO	Bit-Field with capabilities of an attached dome <0>: AZ rotation <1>: has slit <2>: has flap <3>: has seal <4>: is roll-roof <5>: is clam-shell or two part roll-roof
DOME.DIAMETER	FLOAT	RO	Dome diameter.

8 Module LOCAL

This module shall provides information about the (geographical) telescope position, the current time in various formats. This module shall provide the client with the time base that the telescope uses internally for trajectory execution.

Name	Type	Access	Description
UT1	FLOAT	RO	Precision time source with millisecond resolution. This is the time used internally for executing trajectories. The format is seconds since January 1st, 1970.

Name	Type	Access	Description
UTC	FLOAT	RO	Precision time source with millisecond resolution.
TAI	FLOAT	RO	Precision time source with millisecond resolution.
UT1-UTC	FLOAT	RW	Offset between UT1 and UTC (as of 7 Apr 2004 -0.44 seconds). Will return immediately.
TAI-UTC	FLOAT	RW	Leapseconds between TAI and UTC (as of 1998 32.0). Will return immediately.
LATITUDE	FLOAT	RW	Latitude in degrees. Will return immediately.
LONGITUDE	FLOAT	RW	Longitude in degrees, counts positive towards west. Will return immediately.
HEIGHT	FLOAT	RW	Height above sea level in meters. Will return immediately.
SYNCPOS	INT	RW	Sync position with external sources. Will return immediately.

Note: For recent values of UT1-UTC and TAI-UTC see ERS Rapid Service / Prediction center, homepage http://maia.usno.navy.mil.

If SYNCPOS is $\neq 0$, LATITUDE, LONGITUDE and HEIGHT shall be set by external sources, if available (e.g. a GPS receiver). They can also be set by the client for testing purpose (setting one of these variables shall automatically set SYNCPOS to 0).

9 Module COVER[]

This module (or, optionally, module array) provides functions to move covers, e.g. mirror covers. It shall be a generic axis without a TRAJECTORY sub-module and the unit is unspecified.

Note: Depending on the cover device, CURRSPEED and CURRACC might not provide valid data or may not be used at all.

10 Module DOME[]

This module array provides functions to move a telescope dome enclosure. It is implemented as an array of generic axes. It may feature a `TRAJECTORY` sub-module to allow time synchronized dome movements (most likely, this will only be available for the azimuth rotation axis).

Depending on the dome type, the meaning of the axis indices changes. To detect the kind of the dome hardware and the dome features, the `CONFIG.DOME.CAPABILITIES` variable can be used. The next table gives an overview of the used indices for several dome types:

Dome type	Axis index
Fold enclosure	1: open/close
	3: seal (optional)
Roll-Roof	1: open/close
	3: seal (optional)
Clamshell	1: open part 1
	2: open part 2
	3: seal (optional)
Classic astro dome	0: AZ-rotation
	1: slit (optional)
	2: flap (optional)
	3: seal (optional)

Note: Depending on the dome device, `CURRSPEED`, `CURRACC` and `CURRJRK` might not provide valid data or may not be used at all.

11 Module GPS

With this module, access to the GPS receiver is possible. Time and position information can be acquired. The following functions are provided in this module:

Name	Type	Access	Description
`TIME`	FLOAT	RO	Date and time in seconds since 1.1.1970 00:00.
`POS`	STRUCT	—	Structure with the GPS Position (see below).

The GPS.POS structure gives access to the GPS position from the receiver.

Name	Type	Access	Description
LAT	FLOAT	RO	Latitude in degrees.
LON	FLOAT	RO	Longitude in degrees.
HEIGHT	FLOAT	RO	Height in meters.

12 Module MIRROR

This module provides functions to move a mirror (most commonly M3) to select between different foci of the telescope. It is a general axis without a TRAJECTORY sub-module. The positioning of the mirror is discrete and the coordinate system used is the same as for the derotator positions. So a specific optical port can be selected by simply setting the mirror target position to its index (see appendix C). **Note**: Depending on the mirror device, CURRSPEED and CURRACC might not provide valid data or may not be used at all.

13 Modules SENSOR[] and SWITCH[]

These module arrays provide access to all temperature, pressure etc. sensors and to all switchable components of the telescope. These are of course strongly hardware dependent, therefore just a generalized interface is given:

Name	Type	Access	Description
NAME	STRING	RO	Sensor/Switch name.
DESCRIPTION	STRING	RO	A textual description of the sensor.
UNIT	STRING	RO	The unit of the sensor/switch device like Pa, °C etc.
VALUE	INT	RO/RW	Sensor value in the specified unit (RO), Switch value in the specified unit (RW). Returns immediately.

14 Module POINTING (optional)

Since OpenTCI features trajectory support for all axes, there is actually no need for an implementation to include sophisticated astronomical pointing code but let any user / software use it's trajectories to point any object with any speed / positions. However, it is sometimes convenient (e.g. for testing) to have direct pointing functionality in the OpenTCI itself. This module provides a simple interface to point the

telescope to sidereal RA/DEC positions. It is limited by design in the sense that:

- it can only track extra-solar objects;
- it cannot take proper motions, radial velocities etc. into account;
- thus it may only track for a short time with sufficient precision;
- it will only accept J2000.0 coordinates.

The following functions are provided in this module:

Name	Type	Access	Description
CURRENT	MODULE	RO	Current telescope position in equatorial J2000.0 coordinates.
TARGET	MODULE	—	Target position in equatorial J2000.0 coordinates.
TARGETDISTANCE	FLOAT	RO	Distance of all controlled axes from the target.
SLEWINGTIME	FLOAT	RO	Time until telescope will reach target.

Name	Type	Access	Description
TRACK	INT	RW	Start or stop tracking on RA/DEC position given in TARGET sub-module. Possible values are: 0 = stop tracking 1 = start tracking with normal orientation, i.e. positive zenith distance (AZ-ZD) or normal declination (RA-DEC) 2 = start tracking letting the software select a suitable orientation (if the telescope is capable of both orientations); for AZ-ZD the software will chose a orientation that avoids the other axis to run into a limit switch, for RA-DEC the softeware will chose a orientation depending on axes min/max values and the position of the target with respect to the meridian 3 = start tracking with reverse orientation, i.e. negative zenith distance (AZ-ZD) or inverse declination (RA-DEC), 4 = select orientation automatically to minimize slewing time. After writing 2 or 4 the variable will either be 1 or 3 on readout, indicating which orientation was selected. Starting the tracking (i.e. by setting a value $\neq$ 0 and the previous value was 0 will only return if the tracking has ended. Subsequent sets will return immediately. Additionally, some flags can be added to the track mode: 128: sync connected dome 256: sync connected derotator
DOME_DEVIATION	FLOAT	RW	When tracking with dome, maximum allowed deviation between dome target position and current dome position.
POINTINGPARAMS	STRUCT		Contains pointing model coefficients.
REFRACTION	STRUCT		Refraction correction settings.

14.1 Variables of TARGET and CURRENT

The CURRENT modules gives access to the actual (current) telescope position in equatorial J2000.0 coordinates whereas the TARGET module specifies the coordinates, the telescope should reach via setting of the TRACK variable. Unless denoted otherwise, variables of the TARGET module are writable, variables of the CURRENT module are read only. During tracking both modules should return nearly identical values and the difference between the two can be used to determine the tracking performance.

Name	Type	Access	Description
RA	FLOAT	RW	Right ascension. Returns immediately.
DEC	FLOAT	RW	Declination. Returns immediately.
HA	FLOAT	RO	Hour angle.
PA	FLOAT	RO	Position angle.
DA[1]	FLOAT	RO	Target pointing correction in azimuth.
DZ[][1]	FLOAT	RO	Target pointing correction in zenith distance.
DH[2]	FLOAT	RO	Target pointing correction in hour angle.
DD[2]	FLOAT	RO	Target pointing correction in declination.
SIDEREAL_TIME	FLOAT	RO	Current sidereal time.

[1]) available on AZ-ZD and ZD-ZD mounts. Array of size 2 in the latter case, plain variable else.
[2]) only available on RA-DEC mounts.

14.2 Pointing model parameters

This structure holds pointing model coefficients. Unless noted otherwise, the units are degrees. The used pointing correction model should be the standard geometrical model with 6 coefficients for mounting errors plus one tube flexure term. Implementations are however free to implement own models as it suits them. However they must provide at least the following functionality to calculate a model:

Name	Type	Access	Description
DUMPFILE	STRING	RW	Filename of a measurement dump file. Returns immediately.
LOADFILE*	STRING	WO	Loads a previously created dump file to resume a pointing model creation. Returns immediately.

Name	Type	Access	Description
SAMPLE	INT	WO	Write 1 here to take a measurement. If a DUMPFILE is given, the measurement must also be saved to that file. Returns immediately.
CALCULATE	FLOAT	RW	Calculate a model and fill all parameters with the new model on write. Possible values are: 1 = calculate, 2 = calculate and reset any axis OFFSET to 0. Returns fit quality on read. Will only return if the calculation is finished. Depending on the number of measurements, this may take some time.
RECORDCOUNT	FLOAT	RW	Number of samples taken so far. Erase all samples on write. Returns immediately.
LIST*	STRING	RO	Comma separated list of saved pointing models.
LOAD*	STRING	RW	Load a previously saved model by a given name. Returns immediately.
SAVE*	STRING	WO	Save current model to a given name. Returns immediately.
ERASE*	STRING	WO	Erase a previously saved model.

*) Optional, needs not to be supported.

The POINTING module must keep the telescope pointing continuously to positions that are computed with the following steps:

1. Convert RA/DEC to mount coordinate system
2. apply refraction (if enabled)
3. apply pointing corrections

14.3 Refraction parameters

The following values can be modified to control the refraction compensation:

Name	Type	Access	Description
MODE	INT	RW	0 - no refraction correction, 1 - use defaults for refraction (10 °C, 1010 mbar, corrected by height), 2 - use customized settings. Returns immediately.
TEMPERATURE	FLOAT	RW	Outside temperature in Celsius (for mode 2). Returns immediately.
PRESSURE	FLOAT	RW	Air pressure in mbar (for mode 2). Returns immediately.

A Quick guide for the impatient

In the following table a quick overview is given, what variables are needed to do certain actions.

If you want to...	... use variables:
switch on the telescope	set CABINET.POWER=1, check for CABINET.POWER_STATUS to become 1.
switch off the telescope	set CABINET.POWER=0, check for CABINET.POWER_STATUS to become 0.
figure why something does not work as desired	check CABINET.STATUS.STATUS, CABINET.STATUS.LIST for errors.
position an axis	check <AXIS>.REFERENCED to be 1.0, set <AXIS>.TARGETPOS to position.
open/close the dome	set DOME.TARGETPOS.
stop an axis	set <AXIS>.STOP=1.
point to a sidereal target	set POINTING.TARGET.RA + POINTING.TARGET.DEC, then set POINTING.TRACK, check POINTING.TARGETDISTANCE to become reasonably small.
center the target	set <AXIS>.OFFSET.
create a pointing model	set <AXIS>.OFFSET to center the target, set POINTING.POINTINGPARAMS.SAMPLE=1 to add measurement, check POINTING.POINTINGPARAMS.RECORDCOUNT for number of measurements made so far; set POINTING.POINTINGPARAMS.CALCULATE to calculate model.
save a calculated model	set POINTING.POINTINGPARAMS.SAVE=<name>.

If you want to...	... use variables:
load a calculated model	set POINTING.POINTINGPARAMS.LOAD=<name>.

B TPL2 Error and Event numbers

In the following table a generic axis is abbreviated with **A**, the cabinet with **C**, the cabinet's status module with **S** and a generic axis' trajectory with **T**. Events will be written in capital letters. User defined events and errors should start from 1000.

Name	Value	Module	Description
unknown	1		Unknown error.
libfail	2		A hardware library caused an error.
confail	3		Communication error with telescope hardware.
nocon	4		No communication with telescope hardware.
invmode	5	A, C	Telescope is in invalid operation mode.
localmode	6	A, C	Telescope can only be operated manually.
timeout	7	A	Timeout while positioning.
limit	8	A	Limit reached.
softlimit	9	A	Software limit reached.
targetchange	10	A	Target was changed.
nopoints	11	T	Trajectory cannot start (no data).
nobuffer	12	T	No more trajectory buffers.
internal	13		Internal errors.
LATCHCHANGE	14	C	TCS state change.
ERRORCHANGE	15	A, C	Errors have changed.
MANUALCONTROL	16	C	Manual Control setting changed.
MODECHANGE	17	C	Cabinet operation mode was changed.
REFERENCE_EXECUTE	18	A, C	Component has started to reference.
REFERENCE_GOOD	19	A, C	Component is referenced.
REFERENCE_BAD	20	A, C	Component is not referenced.
POWERCHANGE	21	A, C	Component's power has changed.
LIMITCHANGE	22	A, C	Component's limit switch state changed.
MOTIONCHANGE	23	A, C	Component's movement state changed.
POSITION_LOOSE	24	LOCAL	GPS Position is uncertain.
POSITION_GOOD	25	LOCAL	GPS Position is good.
TIME_LOOSE	26	LOCAL	Clock is uncertain.
TIME_GOOD	27	LOCAL	Clock is good.
LEAPSEC	28	LOCAL	A leap second was inserted.
TRAJ_GOOD	29	A	Trajectory runs and accuracy good.
TRAJ_LOOSE	30	A	Trajectory runs but not accurate.

Name	Value	Module	Description
TRAJ_LOWPOINTS	31	A	Trajectory almost at end.
TRAJ_STOP	32	A	Trajectory ended.

C Coordinate Systems of the Telescope Axes

An overview of the coordinate systems used on all telescope axes is given in fig. 1 and 2.

C.1 Azimuth, Zenith Distance and Focus

Azimuth positions are given in geodetic coordinates: 0° is north and east counts positive. Zenith distance positions count positive towards a telescope that is pointing north (i.e. 0° azimuth position). Focus positions are given in millimeters and 0 mm is the innermost position that M2 can reach, counting positive away from M1.

C.2 Derotators, Filters and M3 rotation

Possible derotator locations at different Foci are indexed as follows: 0: Cassegrain; 1: at M2; 2: Nasmyth 1; 3: Nasmyth 2; 4, 5, 6, 7, 8, 9, 10, 11: starting from top of the tube at ZD=90°, AZ=0°, counting locations anti-clockwise in 45° steps.
Every derotator position is counted anti-clockwise. 0° is the position at top where a camera is oriented straight up.
Nasmyth 1 is at ZD=90° from top view the left focus, Nasmyth 2 the right focus.
Note: The difference between index 6/2 and 10/3 is that the Nasmyth derotators are fixed at the bearing retainer and the 6/10 derotators are fixed at the tube,
Filters are indexed by the same scheme.
The M3 mirror rotation uses the same indices to move M3 to the different optical ports.

C.3 Covers and Domes

The mirror cover and dome (which can be thought of as a large cover) positions are 0 if the respective protective cover is closed and non-zero if they are not closed.
On astro-domes (which is represented as axis with array'ed position variables) index 0 represents the azimuth position of the slit in the same coordinates as the azimuth itself, index 1 represents the slit-width and again is 0 if closed, non-zero if opened.
On fold enclosures only index 0 is present with position 0 closed and 180 completely open.

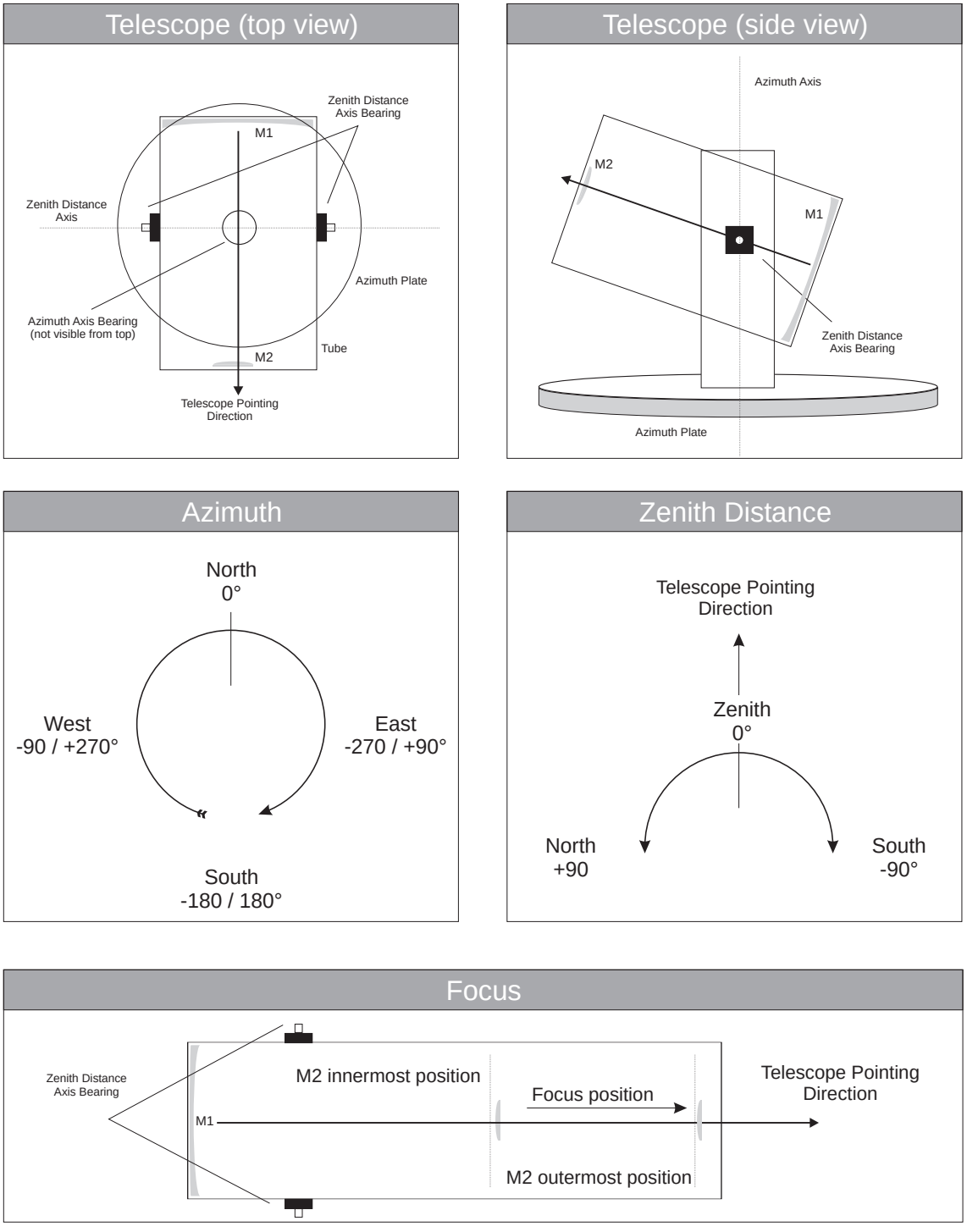

Figure 1: *Overview of the coordinate systems*

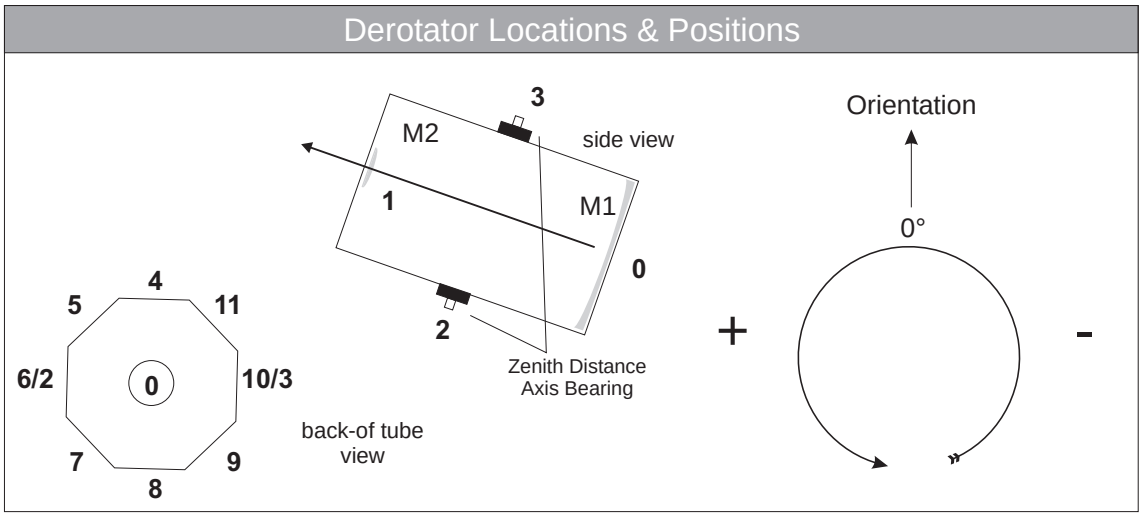

Figure 2: *Overview of the derotator coordinates*

On clamshell domes index 0 and 1 represent both shells and 0 is closed, 90 is completely open.

C.4 Other coordinates

The `GPS` and `LOCAL` Module provide position and time data. Again, latitude and longitude are given in geodetic coordinates: latitude 0° is the equator, 90° is the north pole, -90° the south pole, longitude 0° is Greenwich counting positive towards east. The height is given in meters, counted positive from sea-level.
Temperatures are given in degree Celsius, pressures are given in hPa / mbar.
Times are usually given in seconds since unix epoch (1.1.1970 00:00).

D Abbreviations

Abbreviation	Description
TCS	Telescope Control System
TPL2	Transfer Protocol Language V2
OpenTCI	Open Telescope Control Interface

References

[1] M. Ruder and D. Plasa. *TPL2, Transfer Protocol Language, V2 — A protocol for client-server based exchange of data and commands over a TCP/IP network connection.* 4pi systeme GmbH. `4PI-DOC-03-008-01`.