PSCHTCM

A custom server backend for the Piszkés/Schmidt telescope control system

Latest version: `0.3.1` (2011.07.20)

Software and documentation: written and maintained by András Pál (`apal@szofi.net`)

Contents

1 Introduction

This document summarizes the key features and functionalities of the customized server backend software named "PSCHTCM", intended to control the mount of the **90/60/180 Schmidt telescope** (later on, simply "telescope", located on the Piszkéstető Mountain Station of the Konkoly Observatory of the Hungarian Academy of Sciences[1]) and the additional elements of the telescope system: such as the dome motion, the dome slit door, the focuser built in the telescope tube, the flat lights mounted inside the dome and so on.

As of this writing, the "PSCHTCM" server backend is a core part of the telescope control system (TCS) that currently operates the routinely observations performed with the telescope. This document summarizes the layers what are marked in Fig. 1.

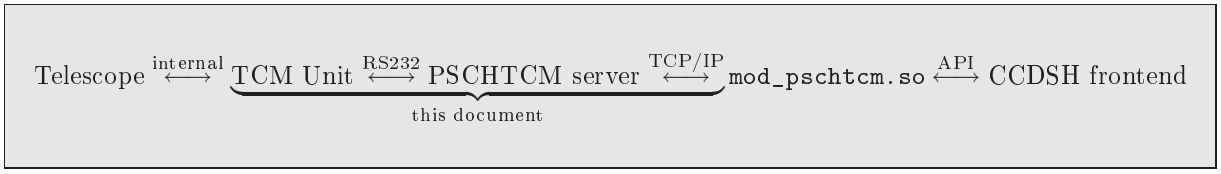

Figure 1: The hardware and software layers of the telescope control system of the Piszkés/Schmidt telescope and the parts of which documentation are covered here.

The structure of this document goes as follows. Section 2 describes the essentials of the TCM unit itself and the RS232 protocol commands that are exploited by the PSCHTCM server. Please note that this description presented in Sec. 2 *is not* complete at all and it is not a replacement of the *official* documentation of the TCM unit. The official documentations (in Hungarian language) of the parts TCM unit are available from László Döbrentei who is the main developer of the TCM hardware. The only purpose of this section is to summarize *those* commands that are used by this PSCHTCM server. Section 3 summarizes the mechanical and engineering properties and constraints of the telescope mount itself, that are relevant in the understanding of the telescope control. Section 4 briefly discusses the concepts of the system architecture used to operate the PSCHTCM server. And last, Section 5 shows the available higher level protocol commands that can be used to control the telescope via the PSCHTCM server.

[1] http://konkoly.hu

2 The TCM unit

This section briefly summarizes the properties of the TCM unit and the serial commands intended to use for communicating with it.

2.1 Serial interface

For communicating with the computer, the TCM unit uses a baud rate of 19200, 8-bit characters with no parity and single stop bit. The serial protocol built on the top of the serial line is a master-slave style protocol. Namely,

- the communication must be initialized by the client (the computer) by sending the appropriate command;
- the TCM unit should answer this command by sending the appropriate response message;
- and unless there is a command initiated by the client, the TCM unit should not send any data at all.

The timeout of the responses depends on the actual hardware component that is asked by the client. Normally, it should be less then some fractions of the second.

We should mention here that the length of the physical media in between the various components of the TCM unit, there might be no response for a certain command or it might happen that the command itself is lost. Therefore, in any client-side implementation for controlling the TCM unit, care must be taken in order to avoid any hardware-level conflicts.

2.2 Hardware components and modules

2.3 Commands employed by the PSCHTCM backend server

This subsection summarizes the serial commands used for controlling the TCM unit. This list of commands is not complete, it contains only the commands that are employed by the PSCHTCM backend server.

As it is implemented in the unit, all of the commands begin with the literal "`#`" character and terminated by a carriage return character. That is, in the ASCII table, is the hexadecimal `0x0D`, `Ctrl+M` or "`\r`" in the C programming language. In the following, where it is necessary, we denote the characters using the C syntax.

2.3.1 Focuser

Commands issued for controlling the focuser begin with "`#A`". The commands that are used by the PSCHTCM backend server to control the focuser are summarized in Table 1.

2.3.2 Equatorial axis

Commands issued for controlling the equatorial axis begin with "`#B`" or "`#F`". The commands that are used by the PSCHTCM backend server to control and track the motion of the equatorial axis are summarized in Table 2.

In order to convert the values of the rotary encoder into hour angles, one should compute

$$\tau = \frac{B}{819.2} - 0.000245, \tag{1}$$

where B is the signed form of the value returned by the command `#BE\r` and τ is the raw (uncorrected) value of the hour angle, in degrees.

Table 1: Serial TCM commands for controlling the focuser. All of the commands begin with "#A", terminated by "\r", while the responses are also terminated by "\r".

Command	Purpose	Response
#A SR\r	Reads the current position of the focuser as returned by the position encoder mounted in parallel with the focal plane.	A real number: the position of the focal plane, in millimeters, terminated by \r.
#A SG <position>\r	Sets the position of the focal plane to the given value of <position>. Like in the case of the #A SR command, the value of <position> is in millimeters.	A single integer, that is 0 (zero) if the command is accepted or non-zero if an error occurred. The response is terminated by \r.

2.3.3 Declination axis

Commands issued for controlling the declination axis begin with "#C" or "#D". The commands that are used by the PSCHTCM backend server to control the motion of the declination axis are summarized in Table 3. In order to convert the values of the rotary encoder into hour angles, one should compute

$$\delta = \frac{C}{4096} - 39.36667, \tag{2}$$

where C is the value returned by the command #CE\r and δ is the raw (uncorrected) value of the declination, also in degrees.

2.3.4 Dome rotation

Commands issued for controlling the rotation of the Schmidt dome begin with "#E". The commands that are used by the PSCHTCM backend server to control the dome rotation are summarized in Table 4.
The equation below shows how the the dome azimuth can be computed from the value returned by the command #EE\r.

$$A = 0.00137906 \cdot (E - 1569177) - 4 \cdot 360.0 \tag{3}$$

Here, A denotes the real azimuth in degrees and E is the actual integer value returned by the command. The value of A should be interpreted as modulo 360. $A = 0$ is the south direction, $A = 90$ is the west direction, $A = -90$ is the east direction and so on.

2.3.5 Dome slit

Commands issued for controlling the dome slit begin with "#F". The commands that are used by the PSCHTCM backend server to control the dome slit are summarized in Table 5.

2.3.6 Dome lights

Commands issued for controlling the dome flat lights also begin with "#F". The commands that are used by the PSCHTCM backend server to control the dome lights are summarized in Table 6.

Table 2: Serial TCM commands for controlling the motion of the equatorial axis. All of the commands begin with "#B" or "#F", terminated by "\r", while the responses are also terminated by "\r".

Command	Purpose	Response
#BE\r	Reads the current position of the rotary encoder mounted on the equatorial axis.	The returned position is an unsigned integer, that must be interpreted as a signed 24-bit integer. The units are is in 1/819.2 degrees. The response is terminated by \r.
#B HS+\r	Starts the coarse motion of the equatorial axis in positive (westward) direction. The speed of the coarse motion is approximately 2.1°/min.	A single zero, terminated by \r, if the command was accepted.
#B HS-\r	Starts the coarse motion of the equatorial axis in negative (eastward) direction.	A single zero, terminated by \r, if the command was accepted.
#B MH\r	Stops the coarse motion of the equatorial axis.	A single zero, terminated by \r, if the command was accepted.
#B M+ <speed>\r	Starts the fine motion of the equatorial axis in positive (westward) direction. The <speed> parameter must be 1, 2 or 3, that yields a speed of 1"/sec, 1′/sec = 0.0166°/sec and 3′/sec = 0.05°/sec, respectively.	A single zero, terminated by \r, if the command was accepted.
#B M- <speed>\r	Starts the fine motion of the equatorial axis in negative (eastward) direction. The <speed> parameter has the same meaning like in the case of the command #B M+ <speed>.	A single zero, terminated by \r, if the command was accepted.
#B MS\r	Stops the fine motion of the equatorial axis.	A single zero, terminated by \r, if the command was accepted.
#B KJ <state>\r	If <state> is 0, the axis driver uses the default sidereal tracking speed. If <state> is 1, the axis driver uses an alternate tracking speed.	The string KJ=0\r or KJ=1\r, if these respective commands were accepted.
#B KJP <speed>\r	If the alternate tracking speed mode KJ=1 is selected (see above), the <speed> indicates the actual tracking speed offset that is used by the motor driver.	The string KJP= <speed>\r\n (with the same value of the speed as it was set by the command itself), if the command was accepted.
#B KJP\r	Asks the previously set tracking speed offset that was given by the most recent call of #B KJP <speed>\r.	The string KJP= <speed>\r\n, where the value of <speed> is the most recent value of the KJP speed.
#F ST <state>\r	Turns off or on the sidereal clock. By default (after the unit powers up) the sidereal clock is on. By setting <state> to 0, the sidereal clock can be turned off. In order to turn on the sidereal clock again, one has to specify 1 for <state>.	A single zero, terminated by \r, if the command was accepted.

Table 3: Serial TCM commands for controlling the motion of the declination axis. All of the commands begin with "#C" or "#D", terminated by "\r", while the responses are also terminated by "\r".

Command	Purpose	Response
#CE\r	Reads the current position of the rotary encoder mounted on the declination axis.	The returned position is an unsigned integer. The unit of the returned declination axis position is in 1/4096.0 degrees and the zero-point is approximately at $\delta = -39.36°$. This value of the declination is smaller that the mount limitations, so the returned value is always positive. The response is terminated by \r.
#D M+ <speed>\r	Starts the coarse or fine motion of the declination axis in positive (northward) direction. The <speed> parameter must be 1, 2 or 3 for the fine motion, that yields a speed of 1"/sec, $\approx 0.03°$/sec and $\approx 0.105°$/sec, respectively. The coarse motion has the <speed> value of 4, and it yields a speed of $\approx 2.1°$/sec.	A single zero, terminated by \r, if the command was accepted.
#D M- <speed>\r	Starts the coarse or fine motion of the declination axis in negative (southward) direction. The <speed> parameter has the same meaning like in the case of the command #D M+ <speed>.	A single zero, terminated by \r, if the command was accepted.
#D MS\r	Stops the motion of the declination axis.	A single zero, terminated by \r, if the command was accepted.

Table 4: Serial TCM commands for controlling the dome rotation. All of the commands begin with "#E", terminated by "\r", while the responses are terminated by "\r\n". Note the difference, other responses (with the exception of the response for "#B KJP ...") are terminated by \r.

Command	Purpose	Response
#EE\r	Reads the current position of the rotary encoder mounted on the motor that rotates the dome.	The returned position is an unsigned integer. See equation (3) how this value can be converted into dome azimuth value. The response is terminated by \r\n.
#E R <direction>\r	Starts or stops the dome rotation. If the value of <direction> is 01, the dome starts to rotate in positive (sidereal) direction. If the value of <direction> is 02, the dome starts to rotate in negative direction, i.e. the dome azimuth decreases. The value of 00 stops the dome rotation.	Returns 0\r\n, if the command was accepted.

Table 5: Serial TCM commands for controlling the dome slit. All of the commands begin with "#F", terminated by "\r", while the responses are terminated by "\r".

Command	Purpose	Response
#F ST\r	Reads the current status of the dome slit.	The returned value must be interpreted as a hexadecimal number, representing 8 status bits of the dome slit. The response is terminated by \r.
#F O\r	Opens the dome slit.	Returns O\r, if the command was accepted.
#F C\r	Closes the dome slit.	Returns C\r, if the command was accepted.
#F S\r	Stops the motion of the dome slit.	Returns S\r, if the command was accepted.

Table 6: Serial TCM commands for controlling the dome flat lights. All of the commands begin with "#F", terminated by "\r", while the responses are terminated by "\r".

Command	Purpose	Response
#F L<light> <status>\r	Turns on or off a given dome flat light. The value of <light> can either be 1 or 2. The light #1 is the fainter while the light #2 is the brighter one. The <status> should be 1 for turning on the light and 0 for switching it off.	The controller responses 1\r if the command was accepted.

3 Mechanical, engineering, and safety constraints

As it was described earlier, the rotary encoders mounted on the equatorial and declination axes return raw binary position values. These values are transformed into "raw" hour angle (τ) and declination (δ) coordinates using the equations

$$\tau = \frac{B}{819.2} - 0.000245, \tag{4}$$

$$\delta = \frac{C}{4096} - 39.36667. \tag{5}$$

In the following, the constants appearing in the above equations are treated as fixed values. Thus, both the constraints and position corrections are computed using these τ and δ values.

3.1 Movement constraints

The Piszkés/Schmidt telescope has an open fork mount, allowing to slew any celestial positions above the horizon. There is no parts of the sky to which the telescope cannot be pointed to.

The mount itself is a biaxial mount, therefore the spin parity of the underlying $\mathrm{S}_1 \times \mathrm{S}_1 \equiv \mathrm{S}_2 \times \{0,1\}$ equivalence, each celestial position can theoretically be observed in two positions.

The current movement constraints built-in the telescope mount axes are displayed in Fig. 2. The equatorial axis is allowed to freely move between the range of $-120 \lesssim \tau \lesssim 135$ while the the declination axis is allowed to freely move in the ranges where the telescope points upwards. These constraints are guaranteed mechanically by several limit switches mounted on both axes.

The above mentioned limitation in the equatorial axis does not allow to observe most of the celestial targets in two positions.

3.2 Pointing model

As it is known for any mechanical system containing moving parts, the alignment of these moving parts are always not perfect at a certain level. In the case of a fork mount, such misaligned parts are

- the equatorial axis, that might not exactly be parallel with the Earth's polar axis;
- the declination axis, that might might not exactly be perpendicular to the equatorial axis;
- and the optical axis (or tube axis), that might not exactly be perpendicular to the declination axis.

In order to use a better pointing model for the telescope instead of the raw hour angle and declination values (as yielded by equations 4 and 5), a simple pointing model is used in the PSCHTCM backend. This pointing model assumes an isotropic environment for the telescope and neglect any kind of hysteresis effect. Since the placement of the telescope is nearly ideal, the constants quantifying the above listed misalignments would be small (in other words, when these q_i quantities are expressed in radians, then, $|q_i| \ll 1$). It is easy to show that under the above assumptions (isotropic environment and neglecting hysteresis), the corrections for these misalignments can be written as

$$\underline{\underline{H}} \begin{pmatrix} c_\tau & -s_\tau & 0 \\ s_\tau & c_\tau & 0 \\ 0 & 0 & 1 \end{pmatrix} \underline{\underline{X}} \begin{pmatrix} c_\delta & 0 & -s_\delta \\ 0 & 1 & 0 \\ s_\delta & 0 & c_\delta \end{pmatrix} \underline{\underline{T}} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \underline{\underline{\Omega}} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{6}$$

In the above equation, the terms denote the following.

- $(c_\tau, s_\tau) = (\cos\tau, \sin\tau)$, $(c_\delta, s_\delta) = (\cos\delta, \sin\delta)$, where τ and δ are the unprocessed values returned by the rotary displacement encoders mounted on the hour angle (τ) and declination axes (see also the introduction of Sec. 3, how τ and δ are computed from the raw binary values returned by the encoders, see equations 4 and 5).

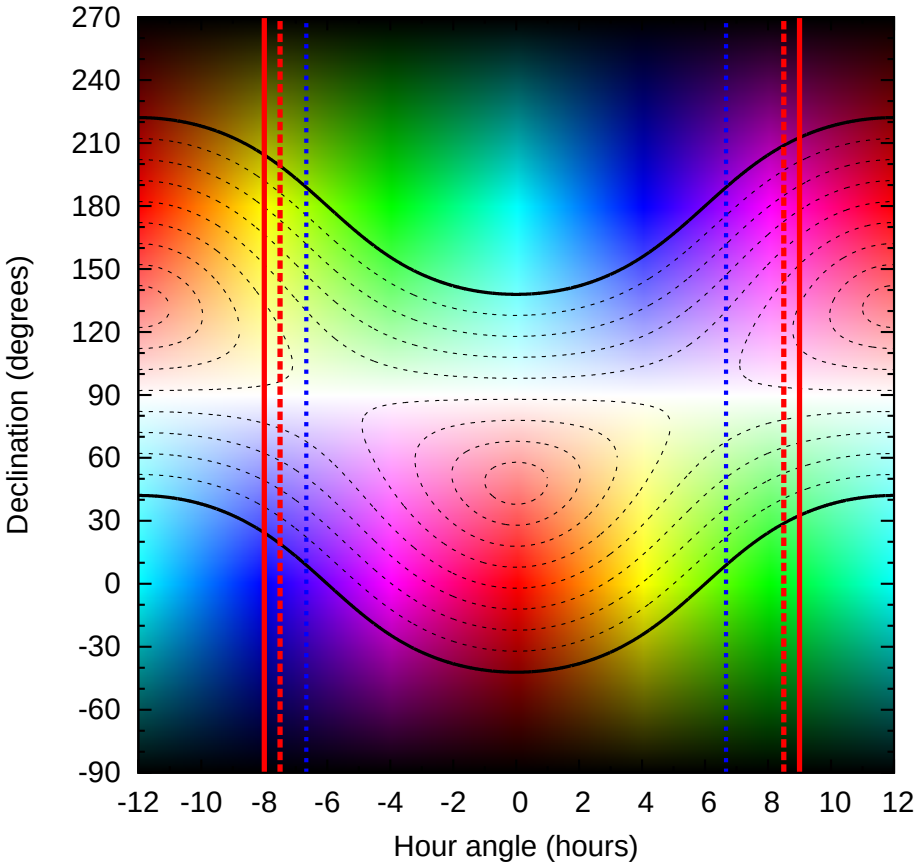

Figure 2: The sky in the first equatorial system, as it is seen by the telescope mount. The hue levels (in parallel with the horizontal axis) represent the celestial hour angle values while the luminance levels correspond to the celestial declination values. The distance in the color space is equivalent to the apparent celestial distance while the projected distance in this graph is proportional to the movements of the equatorial and declination axes. The clear white stripe is the distorted projection of the north celestial pole while the fully black stripe (at the bottom and the top) is the distorted projection of the south celestial pole. The horizon is shown by the two parallel thick black curves while the various positive horizontal elevation levels (in steps of 10°) are shown as thin black dashed curves. The limitations of the motion of the equatorial axis are shown by the vertical red and blue lines. The two thick solid red lines are the hard hardware limitations, beyond which the telescope mount cannot move at all. The thick dashed red lines represent the hard-wired software limitations, which if crossed during sidereal tracking, the track would stop (and the mount goes in an "error" state). The thin dashed blue lines represent the soft limits of choosing "pier-side". During a slew operation, the telescope always choose the telescope axis positions to be in the region limited by the blue lines, and, if this selection is ambiguous, the driver will prefer the point of which declination is smaller than 90°. The apparent sidereal rotation of the celestial sphere is simply a horizontal left-to-right flow on this graph.

- The values (x, y, z) are the J2000 celestial coordinates of the field center, corrected for the aberration and refraction. Namely, from the right ascension α_c and declination δ_c values, the vector (x, y, z) is computed as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\delta_c \cos\alpha_c \\ \cos\delta_c \sin\alpha_c \\ \sin\delta_c \end{pmatrix}. \tag{7}$$

- The misalignments are quantified by the orthogonal $SO(3)$ matrices $\underline{\underline{H}}$, $\underline{\underline{X}}$ and $\underline{\underline{T}}$. Since the mount placement is nearly ideal, these matrices are close to unity.

- The displacement of Earth (or, to be more precise, the equatorial axis of the telescope) in the J2000 reference frame is described by the matrix $\underline{\underline{\Omega}}$. In principle, this matrix contains the effects of the sidereal rotation of Earth, the precession, and the location of the telescope mount on Earth is also encoded in this matrix. The value of this matrix is known in advance.

Since equation (6) does not take into account the field rotation and the matrices and the rotation matrices between $\underline{\underline{H}}$, $\underline{\underline{X}}$ and $\underline{\underline{X}}$, $\underline{\underline{T}}$ in equation (6) have two different (in practice, perpendicular) real eigenvectors, this equation yields 6 independent constants that quantify unambiguously the matrices $\underline{\underline{H}}$, $\underline{\underline{X}}$ and $\underline{\underline{T}}$. Namely, if these matrices written as

$$\underline{\underline{H}} = \exp\begin{pmatrix} 0 & -c & +b \\ +c & 0 & -a \\ -b & +a & 0 \end{pmatrix}, \quad \underline{\underline{X}} = \exp\begin{pmatrix} 0 & -f & +e \\ +f & 0 & -d \\ -e & +d & 0 \end{pmatrix} \quad \text{and} \quad \underline{\underline{T}} = \exp\begin{pmatrix} 0 & -r & +q \\ +r & 0 & -p \\ -q & +p & 0 \end{pmatrix}, \tag{8}$$

these 6 independent constants will be a, b, $c+f$, d, $e+q$ and r. The best-fit values of these constants that are currently employed in the pointing model of the PSCHTCM server are

$$\begin{aligned} a &= -90 \pm 34 \\ b &= -544 \pm 32 \\ c+f &= -1253 \pm 178 \\ d &= -56 \pm 31 \\ e+q &= 4 \pm 143 \\ r &= 448 \pm 196 \end{aligned}$$

where the values given in this table are in micro-radians. These values are then used to be substituted into equation (6) in order to conver the raw hour angle and declination values (as read by from the rotary encoders, see equations 4 and 5) to the corrected coordinates. By inverting equation (6), it is possible to slew the telescope to any given position within the accuracy of the pointing model. All in all, this accuracy of the pointing model discussed here is currently $\approx 10" - 15"$.

4 Architecture concepts

Actually, the PSCHTCM software is designed to run on any Linux-based hosts that supports at least a single serial port (attached to a device node, e.g. `/dev/ttyS0`) and with the capability of TCP/IP networking. There is no further requirements on the running hardware. However, the only relevant thing might be that the precise and accurate timing of the telescope movement might require additional installations of a time synchronization daemon, for instance, an NTP client.

By default, the PSCHTCM software runs in a server-mode, allowing clients to connect to its port. The default listening TCP port of the PSCHTCM server is 8873, but this can be altered via the command line argument `-p` or `--port` (and in addition, the server socket can also be bound to local UNIX sockets as well). The PSCHTCM program can also be started in "interactive mode", using the command line arguments `-i` or `--interactive`. In this case, PSCHTCM does not allow any remote clients to connect, instead, it opens a single channel for communication and use the standard input/output for retrieving the commands and sending the appropriate answers. This "interactive mode" allows the developers or maintainers to test safely the functionalities of the server and also useful for debugging.

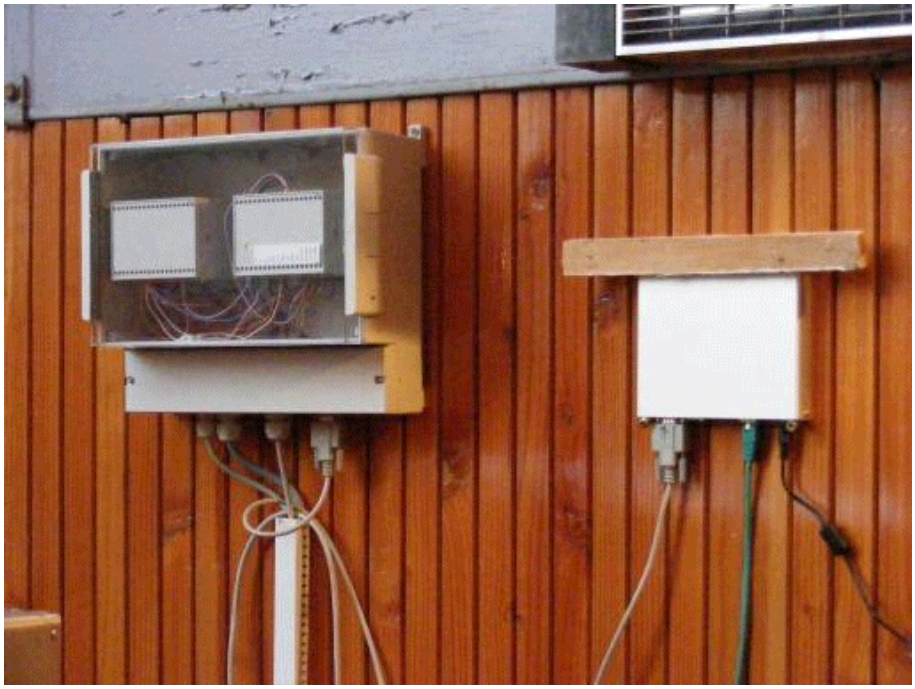

Figure 3: A picture showing the TCM unit connectors (on the left) and the ALIX computer currently running the PSCHTCM server (on the right). Both boxes are mounted inside the dome, close to the main controller of the telescope.

As of this writing, the PSCHTCM server backend runs on an embedded-class computer with no moving parts. This computer utilizes a PC Engines[2] manufactured board ALIX.6E1[3], put in a small aluminium chassis with the size of approximately $18 \times 18 \times 2$ centimeters and with connectors on the bottom side (see also Fig. 3). The whole operating system, a customized Debian GNU/Linux "Lenny" (vers. `5.0.2`) with the kernel `2.6.30-486-voyage` is installed on a 4 GB CF card. The CF card has a single partition with an `ext2` file system (i.e. without any internal or external journalling) and this file system is mounted read-only during the whole operation.

This main board (ALIX.6E1) has a single serial port (seen by the kernel as `/dev/ttyS0`): it is used to connect the PSCHTCM server to the serial connector of the TCM unit. This board has two ethernet controllers from which only one (the `eth0` interface) is used now.

[2] http://www.pcengines.ch/
[3] http://www.pcengines.ch/alix6e1.htm

5 The protocol and available commands

This section describes the commands available to the clients connecting to the PSCHTCM backend servers. In principle, these commands implement a master-slave protocol in a polling fashion. In other words, all of the commands are expected to response promptly, and, depending on the actual command, subsequent polling is required to check if the desired operation has been completed or not.

Both the commands and the responses are single ASCII lines, i.e. strings terminated by the UNIX newline character (`0x0A`, Ctrl+J, or "`\n`" in the notation of the C language). Each command is one or more space-separated token, while the response is always an integer status code, followed either by a status descriptor or the optionally requested data (these tokens are separated by spaces). The status descriptor is a single, uppercase word that is unique for each status code (i.e. each status code have a corresponding status descriptor, but there might be similar status descriptors for different codes). For instance, to turn on light #1, one has to issue the command "`light 1 1\n`" while the response will be "`100 OK\n`", indicating that the command has been accepted and processed. The status codes has the following values:

- Codes between 100 and 199 indicate a successful acknowledgement of the command. Usually, codes 100 and 101 are followed by the status descriptor "`OK`" (see the above example) or the requested data.
- Codes between 200 and 299 indicates an error, both on the client side or in the server side (including hardware failures). For instance, the code 201 is for invalid command (thus, the command sent to the PSCHTCM server is not valid, therefore completely ignored), and has a status descriptor "`ECMDINVALID`".
- Codes between 300 and 399 indicates warnings. In this case, there is no effect of the command, but in other circumstances, the command might be accepted in the similar form. For instance, if a slewing request would move the telescope below the horizon, the response code is 301 and the descriptor is "`WBELOWHORIZON`".

In the following, the available commands are detailed. Since each command is terminated by the newline character and each response is also a single line, terminated by newline, below we do not indicate the trailing "`\n`" characters.

5.1 Status commands

Command syntax: `mountstatus`

This command returns the current status of the mount. The status descriptor is followed by an integer, that is:

- 0, if the mount is ready to use.
- a positive number, if the mount is busy with some operation, and
- a negative number if some hardware level error is occurred.

This integer status is followed by a status descriptor, that can be `idle` if the mount is ready to use, `slewing` if the telescope mount is in motion, and `unreachable` if there is no response from the TCM unit. If the mount is ready or busy (i.e. the status is `idle` or `slewing`), then the status descriptor is followed by the uncorrected (raw) values of right ascension, declination, local sidereal time, and hour angle. All of these quantities are expressed in degrees (including the right ascension and local sidereal time as well!). These quantities are followed by the motion status of the mount.

Command syntax: `domestatus`

This command returns the current status of the dome. The status descriptor is followed by an integer, that is:

- 0, if the dome is ready to use.
- a positive number, if the dome is busy with some operation, and

- a negative number if some hardware-level error is occurred.

This integer status is followed by a status descriptor, that can be `idle` or `tracking` if the dome is ready to use, `rotating` if the dome is in motion, and `unreachable` if there is no response from the TCM unit.

Command syntax: `slitstatus`

This command returns the current status of the dome slit. The status descriptor is followed by a real number. This is negative if there is an error with the dome slit (the position cannot be obtained or there is no communication with the TCM unit). If the slit is completely closed, this number is 0, if it is completely opened, the number is 1, if the slit is partially opened, this number is somewhere between 0 and 1, according to the actual position of the slit door. This number is followed by the hexadecimal status code of the dome slit (see also the "`#F ST\r`" TCM command), and a status descriptor (that can be `unreachable`, `stucked`, `closed`, `opened`, `partial` and `unknown`).

Command syntax: `focusstatus`

This command returns the current status of the focuser. The status descriptor is followed by an integer, that is:

- 0, if the focuser is ready to use.
- a positive number, if the focuser is busy with some operation, and
- a negative number if some hardware-level error is occurred.

This integer status is followed by a status descriptor, according to the status code. It can be `idle` if the focuser is ready to use and it is not in motion, `moving` if the focuser is currently moving to a previously set position, and `unreachable` if there is no response from the TCM unit.

5.2 Locking

Command syntax: `lock`

This command asks the lock for the client. If the lock cannot be acquired, the command fails, otherwise the client grabs the lock until connected to the server or explicitly releases the lock by the `unlock command`. Note that this locking mechanism is always non-blocking, so the client receives an error (`ELOCKED`) if another client has already acquired the lock.

Locking both disables other clients to successfully issue the `lock` comment and reduces the available commands to the passive ones (i.e. the commands `slew`, `domemove`, `dometrack`, ... will also yield a response of `ELOCKED`).

Command syntax: `unlock`

This command releases the lock for the client, or does nothing if noone else has grabbed a lock. If the lock is owned by the other client, the server responses with an error of `ELOCKED`.

5.3 Mount control

Command syntax: `mountposition`

This command returns the corrected mount position. If the readout of the mount coordinates are successful, the response contains the corrected coordinates of right ascension (α_c), declination (δ_c), field rotation and local sidereal time. All of these values are in degrees.

Command syntax: `mountlimit`

This command returns the current limitations of a freely tracking mount. The status code is followed by a number and a descriptor. If the returned number is positive, the mount can freely track the current position for the time given by the number. If the number is negative, the mount is currently exceeding the limits and possibly halted (does not do any sidereal or other kind of tracking). The descriptor can either be `taulimit` or `horizon` if the limitation will be exceeded due to the limitations in the equatorial axis or due to the horizon, respectively. If the number is negative (practically, -1), the descriptor can be `tauexceeded` or `belowhorizon`, indicating an overrun of the equatorial axis or the mount points below the horizon.

Note that the free tracking time is returned in synodic degrees. Use known the conversions to convert into seconds, minutes or hours.

Command syntax: `slew ra=<RA> dec=<DEC>`

This command sends the mount to the desired position. The right ascension and declination values are always for the current epoch and the mount position corrections are also taken into account. Both the `<RA>` and `<DEC>` values should be given in degrees. The traditional sexagesimal notations are not accepted here.

Use the command `mountstatus` to check if the slewing has been finished.

The command will return a warning (code 301, response `WBELOWHORIZON`) if the target coordinates point below the mount limitations. In this case the whole slewing request is ignored.

Command syntax: `stop`

This command stops any movement of the telescope mount instantly, with the exception of the sidereal tracking. Note that due to the inertia of the mount, the telescope tube will stop in a few seconds after issuing this command.

Command syntax: `mounttrack [0|1]`

This command stops or (re)starts the sidereal tracking of the dome. Giving a parameter of `0` stops, while `1` (re)starts the tracking.

Command syntax: `mounttrackspeed [off|<speed>]`

This command alters the sidereal tracking speed of the equatorial axis. If the value of `<speed>` is 0 or the literal "`off`" string, the mount track speed will be reset to the default (sidereal) speed. Otherwise, the parameter `<speed>` will be interpreted as an offset in the units of degrees/second in the direction of right ascension (thus, inverse hour angle and apparent rotation). If the speed parameter `<speed>` is too small or too large (i.e. that particular speed is not supported by the mount driver), the server returns a status descriptor `EOUTDOMAIN`.

5.4 Dome control

Command syntax: `domeazimuth`

This command returns the current azimuth if the dome slit door, in degrees. See also equation (3) for more details on the interpretation of the returned position.

Command syntax: `domemove <azimuth>`

This command starts to rotate the dome to the given azimuth specified in the argument `<azimuth>`. The value of `<azimuth>` is in degrees. 0° is for south, +90° is for west, −90° is for east, etc.

Command syntax: `domestop`

This command stops both the rotation and the sidereal tracking (see also `dometrack`) of the dome. Note that due to the inertia of the dome, the dome will stop in a few seconds after issuing this command.

Command syntax: `dometrack <tau> <dec>`

This command will initiate a sidereal tracking for the dome, in order to track the celestial position described by the `<tau>` and `<dec>` equatorial coordinates. Both `<tau>` and `<dec>` are in degrees. Note that instead of right ascension, the client should specify a hour angle value.

5.5 Dome slit door control

Command syntax: `slit open|close|stop`

This command opens or closes the dome slit door, or, if the slit door is in motion, it can also explicitly be stopped using this command. The command `slit` must be followed by exactly one term, `open`, `close` or `stop`, according to the previously mentioned respective operations.

The status of the (motion of the) dome slit door can be tracked with the command `slitstatus`, see also Sec. 5.1.

5.6 Dome light control

Command syntax: `light <light> 0|1`

This command turns on or off the given dome light specified in the parameter `<light>`. If the last parameter is `0`, the given light is turned off while if the last parameter is `1`, the light is turned on. As of this writing, there are two independent dome lights mounted on the dome, therefore `<light>` can either be `0` or `1`. Actually, the light with the code `0` is the fainter one, mounted on the left-side of the flat screen while the light with the code `1` is the brighter one, mounted on the right-side of the flat screen.

5.7 Focus control

Command syntax: `focusposition`

This command returns the current position of the focuser. Since the focuser built in the Piszkés/Schmidt telescope has only a single degree of freedom, this command returns a single number: the focuser position in millimeters, relative to an arbitrary (but fixed) offset. The response that follows the status code is a string with the format `focus=<position>` where `<position>` is the actual position, in millimeters. The current range of normal positions is between $25.30 \leq F \leq 25.70$ and it seems to be very stable for different ambient temperatures. The resolution of the returned focus position is 0.01 (millimeters).

Command syntax: `focusset <position>`

This command sets the focuser position that is given in the argument, in millimeters. The normal range of positions is is between $25.30 \leq F \leq 25.70$. In the current setup (with the Optec IFW filter wheel and the built-in E, B, V, R, and I filters), the focus positions F are around $F_{\mathrm{E}} \approx 25.40$, $F_{\mathrm{B}} \approx 25.62$ and $F_{\mathrm{V}} \approx F_{\mathrm{R}} \approx F_{\mathrm{I}} \approx 25.52$.