SteelDrive II Focuser
Technical documentation

20th December 2019 v1.100

Contents

Chapter 1

Functional Overview

The SteelDrive digital motor focuser is used for automated telescope focusing. It may be used standalone as a hand-held device, as well as remotely in conjunction with a computer. It comprises a motor unit which is attached play-free to the mechanical focuser of the telescope by means of a timing belt as well as a controller unit which houses the electronics and does also serve as a hand-held keypad. Controller and motor unit are connected via a standard RJ45 (Cat7) patch cable. Temperature sensors may be connected to each of the units. The sensors are used for the automatic compensation of focus drifts due to changes of the external temperature. The temperature sensors may also serve as the input signal for a PID controlled fast-PWM power output. It can be used to precisely control the temperature of dew-heaters, optical filters, mirrors etc.

1.1 High precision focusing of heavy loads

The high-torque stepper motor is directly engaged with the focuser via a timing belt. This eliminates the backlash (and potential cause of defects) induced by gearboxes which would have to be used with lower power motors. The SteelDrive can quickly move loads up to 8 kg. The default focusing resolution is $\approx 2.3\mu m$ (in conjunction with a **Baader Diamond SteelTrack®** focuser).

1.2 Contactless Homing Sensor

The SteelDrive motor unit features a contactless (wearfree) homing sensor (hall-effect sensor). It is activated by a magnet attached to the moving part of the focuser. It enables precise homing and repeatable approach of saved absolute positions as well as reference of the mechanical endstop. However the SteelDrive is also usable without the homing sensor as the absolute position is kept in a persistant circular eeprom buffer. Even if a power failure occures, the last absolute position is loaded upon restart of the SteelDrive.

1.3 Standalone Temperature Compensation

SteelDrive II provides two sockets for the attachment of digital temperature sensors (one sensor is included, sockets on motor unit and controller unit). It is possible to used multiple (up to 8, use 3-pin audio Y-adapter) sensors in parrallel at one sensor socket, their temperature will be averaged (This enables the measurement the temperature eg. above and below the telescope tube, or inside and outside etc.).
For the automatic temperature compensation, the sensor socket to be used may be selected, or the average temperature all sensors attached to both sensor sockets may be used. The temperature compensation factor can be calculated with the **SteelGoII** software and is saved internally in the

SteelDriveII controller unit. The compensation period and ΔT threshold is also adjustable. Once set, the compensation can then also be used in standalone mode of operation (without a connected computer). When the automatic temperature compensation is enabled, the focuser will do a correction movement every period, if the temperature difference threshold ΔT is exceeded.

1.4 12V/1A PWM Power Output

The SteelDrive features a 12V/1A power output. Dew-heaters up to 10W may be directly attached to the SteelDrive controller (with optional adapter cable). The power can be controlled by the PWM setting. Other uses are also possible. The default PWM value may be set via PC software and also used in standalone mode of operation. (Second board revision enables the use of up to 20W heaters if a stronger power supply is used)

1.5 Standalone PID Controller

A further enhancement of the fast PWM power output is the built-in PID (proportional, integrative, derivative) controller. Each of the two temperature sensors sockets as well as the average of both may be used as the input of the PID controller. It enables the precise temperature control of eg. dew-heaters, optical filters and instruments with ± 0.1 degree accuracy. Controlling the dew heaters temperature little above the dew-point, minimizes temperature gradient induced stress in the telescope structure. This can be automatically set, eg. on bootup the controller will always keep the temperature of a dew-heater a given offset (eg. 4°C) above the ambient temperature.

1.6 Superior built quality

The mechanical parts are made in-house out of black anodized, CNC milled aluminum. High quality electronic components with adequate ratings are used for the electronic boards which are also designed and produced in Germany. The electronic boards have been further protective coated.

1.7 Low power consumption

The motor current is automatically reduced while in holding state (adjustable). The SteelDrive then consumes approximately 0.7 W.

1.8 Intuitive Operation

The intuitive hand control enables the user to quickly change between normal speed, jogging mode (number of steps are also configurable via software) and single step mode. The configuration of limits, saving of and goto a known focus position as well as temperature compensation toggling are also available from the hand controller. The backlight illmuniation of the keypad and status LEDs are dimmable.

1.9 Customizable

The SteelDrive II enables power users to customize a lot of settings from temperature compensation, PID control to the adjustment of stepper motor settings. Users with multiple telescopes can use multiple motor units with only one controller one at a time. The setting for the different telescopes are saved and loaded with the SteelGoII software.

Chapter 2

Standalone handheld usage

The SteelDrive focuser may be used standalone (without connection ot another PC). The functions available in standalone mode are:

- Continuous focuser movement.
- Focuser jogging movement.
- Single step movement.
- Homing/Zeroing
- Save outer limit.
- Save focus position.
- Goto saved focuser position.
- Toggle automatic temperature compensation.
- No limits mode.

2.1 Keypad

The SteelDrive focuser features 4 back-illuminated buttons and two status LEDs (see figure 2.1).

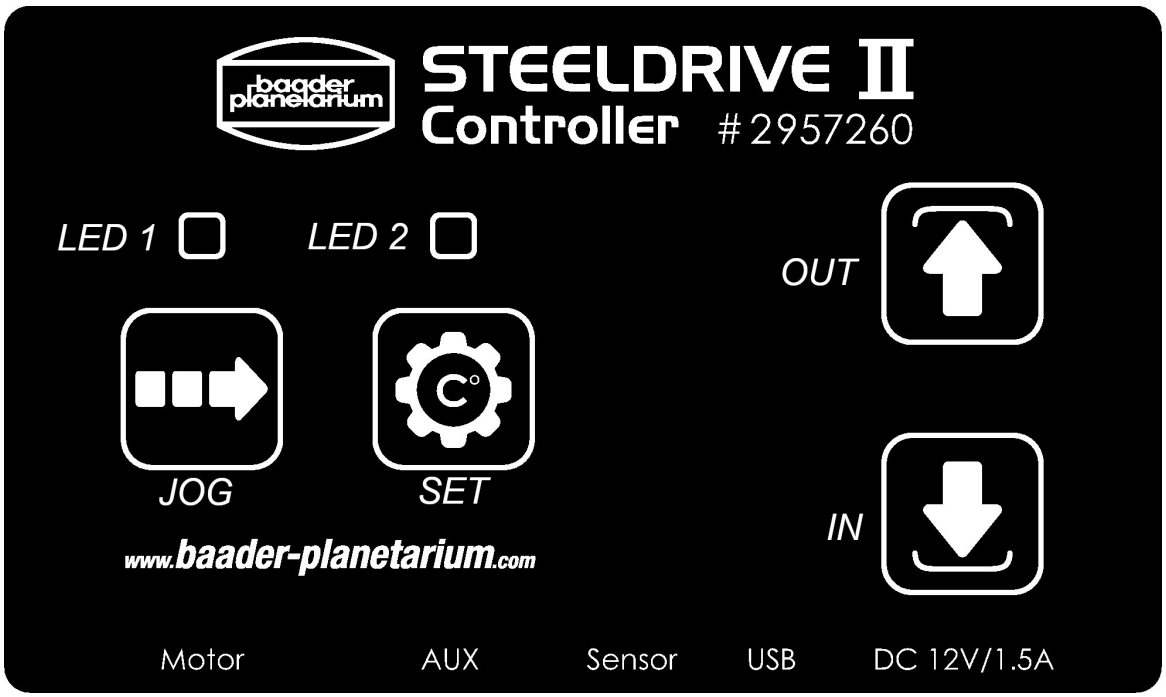


Figure 2.1: SteelDrive keypad

2.2 Modes of operations

The available keypad functions are structured in five modes of operation:

- NORMAL mode
  - Continuous focuser movement
  - Goto saved position
- JOG mode
  - Focuser jogging
  - SINGLE STEP mode
    - Single step movement
- SET mode
  - Zeroing
  - Save limit
  - Save position
  - OVERRIDE LIMITS mode
    - Continuous focuser movement ignoring the limits

1. NORMAL mode: The normal mode is entered when the device is powered. The two status LEDs are off.

2. JOG mode: The jogging mode is entered when the *JOG* button is pressed once. Status *LED1* is lit.

3. SINGLE STEP mode: The single step mode is entered when the jogging mode is active and the *SET* button is pressed. *LED1* and *LED2* is lit.

4. SET mode: The setting mode is entered when the *SET* button is pressed. Status *LED2* is lit.

5. OVERRIDE LIMITS mode: When the *SET* button is held 3s in SET mode, the override limits mode is activated (*LED2* blinking fast). The saved maximum travel limit and the homing sensor are *NORMAL* mode)

2.3 Usage

2.3.1 Setup

Assume the motor unit was mounted to the **SteelTrack Diamond** focuser and connected to the **SteelDrive** controller. One or two temperature sensors were connected.

Zeroing

The first thing is setting a correct zero position.

- If the homing sensor is used (the magnet is attached to the focuser), power on the device by plugging in the attached power supply.
  - If the automatic homing was activated in the SteelGO software, Enter the *SET* mode by pressing *SET* once (*LED2* is lit). Now hold *IN* for 3s. The focuser will then move inwards until the homing sensor is activated, set its position to zero and subsequently enter the *NORMAL* mode again (*LED1* and *LED2* are off).
  - If the automatic homing was no yet activated, but the sensor magnet is attached to the focuser, use the to move the focuser inwards until the home sensor is activated. The focuser will then stop and set the correct zero position.
- If the homing sensor is not used (no magnet attached to the focuser), manually move the focuser to its most inward position. Subsequently power on the device. enter the *SET* mode by pressing the *SET* once (*LED2* is lit). Now hold *IN* for 3s. The device will set its internal position to zero and *NORMAL* mode is entered (*LED2* is off).

Set maximum travel limit

Depending on the whole telescope setup, the desired maximum travel limit may vary. In order to configure a save travel range, move the focuser outward by pressing *OUT* . Once the desired maximum is reached, enter the *SET* mode by pressing *SET* once (*LED2* is lit). Now hold *OUT* for 3s. The maximum travel limit is saved and *NORMAL* mode is entered again (*LED1* and *LED2* are off).
If the limit is to be extended above the current limit, the *NO LIMITS* mode needs to be used (with care!). Enter the *SET* mode by pressing the once (*LED2* is lit). The hold again for 3s until *LED2* is blinking fast. Now the limits are ignored. Travel with and to the desired new limit. Then hold again until *LED2* stops blinking (we are back in NORMAL mode, *LED1* and *LED2* are off). Press again once to enter *SET* mode (*LED2* is lit), then hold for 3s in order to save the new limit.

2.3.2 Focusing

The intended workflow for focusing is:

- Find the approximate focus position by traveling fast in *NORMAL* mode.
- Use the *JOG* mode to limit the range of the best focus to the size of one jogging distance (can be configured, default approx. 0.05 mm).
- Use the *SINGLE STEP* mode to find the best focus position.

Focusing in normal mode

- Press *OUT* to move the focuser outwards as long as the button is held or the outer limit is reached.
- Press *IN* to move the focuser inwards as long as the button is pressed or the zero position is reached.

Use the *NORMAL* mode to find the approximate focus position.

Focusing in jog mode

Press the *JOG* button once to enter the JOG mode (*LED1* is lit):

- Press *OUT* in order to move the focuser for approx. 0.05mm outwards (default), respectively the jogging distance configured in the SteelGo II software.
- Press *IN* to move the focuser approx. 0.05 mm inwards (default), respectively the jogging distance configured in the SteelGo II software.

Use the *JOG* mode to narrow down the range of the optimal focus to one jogging range.

Focusing in single step mode

Assume the JOG mode is active (*LED1* is lit). Press *SET* once in order to enter the *SINGLE STEP* mode (*LED1* and *LED2* are lit).

- Press *OUT* in order to move the focuser one step (approx. 0.027 mm) outwards (with the Baader Planetarium Diamond SteelTrack, the gear-ratio of other focuser may be different).
- Press **IN** to move the focuser approx. 2.275 μm inwards.

Use the *SINGLE STEP* mode to find the best focus.

2.3.3 Save the focus position

Leave the *JOG* or *SINGLE STEP* mode by pressing *JOG* once (*LED1* and *LED2* are now off). Enter the *SET* mode by pressing *SET* once (*LED2* is lit).

- Hold *JOG* for 3s in order to save your current position.

2.3.4 Go to the saved focuser position

In order to move to a saved focus position, ideally zero the device first. Then

- in *NORMAL* mode (*LED1* and *LED2* off) hold *JOG* for 3s. The focuser will now move to the saved position. As observation conditions are never the same, subsequent fine focusing is recommended.

2.3.5 Toggle automatic temperature compensation

The automatic temperature compensation is configured via the SteelGO software.

- Enable: In *NORMAL* mode, hold *SET* for 3s (*LED2* is blinking slowly.)
- Disbale: If the automatic temperature compensation is enabled (*LED2* slowly blinking), hold *SET* in *NORMAL* mode in order to deactivate it (*LED2* is now off).

2.3.6 Override limits

In order to override a set limit, enter the *NO LIMITS* mode, by pressing *SET* once (entering the *SET* mode, *LED2* is lit) and then holding *SET* button again for 3s. When *LED2* is blinking fast, the set limits are ignored while moving with and . Move to your new zero or maximum travel position and hold *SET* again for 3s in order to leave the *NO LIMITS* mode. Now set the new limit by pressing *SET* and then hold *OUT* respectively *IN* for 3s.

2.4 Button function overview

		NORMAL mode	 SET mode	 JOG mode	  SINGLE STEP mode
	**OUT**	Move out	Save outer limit (hold 3s)	Jog out	Single step out
	**IN**	Move in	Zeroing (hold 3s)	Jog in	Single step in
	**JOG**	Enter JOG mode	Save position (hold 3s)	leave JOG mode	Enter NORMAL mode
	**SET**	Enter SET mode / Toggle T-Comp. (hold 3s)	Leave SET mode / Enter/Leave NO LIMITS mode (hold 3s)	Enter single step mode	Enter JOG mode

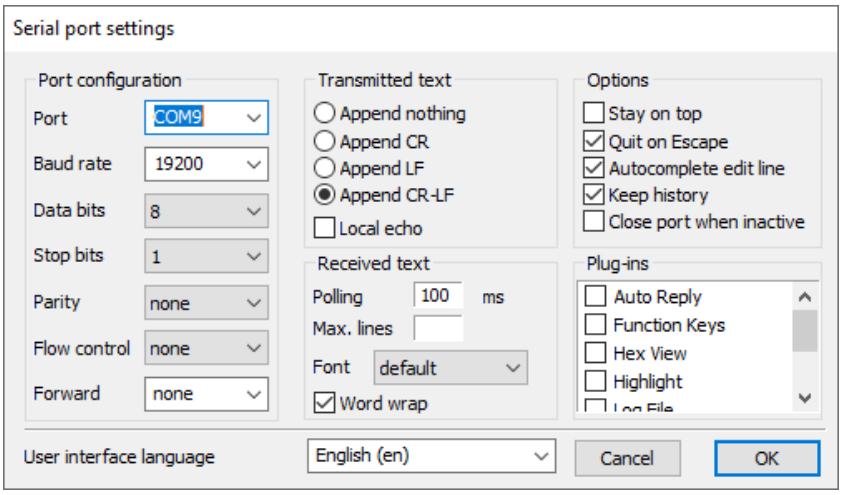
Chapter 3

Communication Protocol

3.1 Virtual COM-Port Configuration

Parameter	Value
baudrate	19200
bytesize	8 bit
parity bit	none
stop bit	one
timeout	0.1s
conoff	false
rtscts	false
dsrdtr	false

Always append CR+LF.



Settings for the RS232 terminal programm Termite

3.1.1 Communication protocol

All characters send to the device will be echoed. This provides an simple way to verifiy the device receives complete messages.

- The device will ignore all essages not starting with the prefix "$BS".
- The communication of the device with the digital temperature sensors involves some blocking operations (up to 500μs). During this period there is a chance that some characters will be missed. Once the device receives any serial communication, it will only communicate with the sensors just before answering to valid serial commands. This is intended for periodically polling (say 1-2 Hz). If there is no serial communication for 10s, the device will poll the sensors every second.
- CRC checksum verification of the communication is not mandatory but encouraged (see 3.2.5).
- On boot, the device will send:
  ```
  $BS Hello World!\r\n
  ```

3.2 General commands

Set variables are persitent after reboot. Often saved variables (like absolute position) are stored in circular EEPROM buffers and only changed bits are updated in order to reduce the buffer wear.

3.2.1 Unknown commands

Unknown/invalid commands are responded with:

```
$BS ERROR: Unknown command!\r\n
```

3.2.2 SET commands

Valid SET commands are responded with:

```
$BS OK\r\n
```

3.2.3 GET commands

Valid GET commands are responded with:

```
$BS STATUS VARIABLE:VALUE\r\n
```

3.2.4 GET VERSION

Get the firmware version. Example Command:

```
$BS GET VERSION\r\n
```

Example answer:

```
$BS STATUS VERSION:0.700(Apr  5 2019)\r\n
```

3.2.5 CRC checksum verification

When checksum verification is enabled, the CRC8 (Dallas/Maxim algorithm, see appendix .1) of the message is calculated, the '*' character (==(uint8t)42) is appended to all messages and then the HEX representation (base 16) of the checksum is appended.
All messages received by the SteelDrive will be verified and ignored on invalid checksum. (except: RESET, REBOOT and CRC_DISABLE)

CRC_ENABLE

Enables CRC checksum verification.
Example Command:

```
$BS CRC_ENABLE\r\n
```

Example answer:

```
$BS OK*21\r\n
```

CRC_DISABLE

Disable CRC checksum verification. The checksum supplied with CRC_DISABLE is not verified. CRC_DISABLE may also be send without attached checksum.

```
$BS CRC_DISABLE\r\n
```

Example answer:

```
$BS OK\r\n
```

3.2.6 REBOOT

Reboot the device.

Example Command:
```
$BS REBOOT\r\n
```
Answer:
```
$BS Hello World!\r\n
```

3.2.7 RESET

Reset the device to factory defaults (except serial number) and reboot.

Example Command:
```
$BS RESET\r\n
```
Example answer:
```
$BS OK\r\n
$BS DEBUG:FACTORY RESET...\r\n
$BS DEBUG: LOADING DEFAULTS...\r\n
$BS Hello World!\r\n
```

3.2.8 NAME

May be used to give the controller a unique name (maximum 19 characters).

SET NAME

Set a device name.

Example command:
```
$BS SET NAME:Focuser1\r\n
```
Example answer:
```
$BS OK\r\n
```

GET NAME

Get the device name.

Example command:
```
$BS GET NAME\r\n
```
Example answer:
```
$BS STATUS NAME:Focuser1\r\n
```

3.2.9 BKLGT

Controls the keypad backlight brightness as well as the brightness of the keypad's status LEDs.

SET BKLGT

Set the BKLGT digital potentiometer whiper setting, controlling the button background and status LED brightness (uint8 [0-100] %).

```
$BS SET BKLGT:100\r\n
```

Example answer:

```
$BS OK\r\n
```

GET BKLGT

Get the BKLGT digital potentiometer whiper setting, controlling the keypad illumination backlight LED brightness as well as the keypad status LED brightness.

Example command:

```
$BS GET BKLGT\r\n
```

Example answer:

```
$BS STATUS BKLGT:100\r\n
```

3.3 Focuser related commands

3.3.1 INFO

Get information about current device state: NAME,POSITION,STATE,LIMIT‡ Useful for polling the current device state. (key-value delimiter = ':', item delimiter = ';') STATE may be GOING_UP, GOING_DOWN, STOPPED, ZEROED. Zeroed means that no movement has been made after that last zeroing. This can be queried by a controlling application after a ZEROING commands has been issued (with USE_ENDSTOP enabled) and the programm is waiting until the automatic zeroing is finished.
Example command:

```
$BS INFO\r\n
```

Example Answer:

```
$BS STATUS NAME:BP_SD_41;POS:497;STATE:STOPPED;LIMIT:25000\r\n
```

3.3.2 SUMMARY

Get more information about current device state:
NAME, POSITION, STATE, LIMIT, FOCUS, TEMP0, TEMP1, TEMP_AVG, TCOMP, PWM
Useful for polling the current device state. (key-value delimiter = ':', item delimiter = ';')
Example command:

```
$BS SUMMARY\r\n
```

Example Answer:

```
STATUS NAME:BP_SD_41;POS:497;STATE:STOPPED;LIMIT:14500;FOCUS:3323;
TEMP0:22.45;TEMP1:21.78;TEMP_AVG:22.12;TCOMP:0;PWM:100\r\n
```

3.3.3 GO

Go to absolute position (if within set limits [0;LIMIT]). If the given position is outside the limits, the focuser will move to the appropriate next limit position.

Example command:

```
$BS GO 1234\r\n
```

Example Answer:

```
$BS OK\r\n
```

3.3.4 STOP

Stop focuser movement immediately. (the focuser will use a deceleration ramp)

Example command

```
$BS STOP\r\n
```

Example Answer:

```
$BS OK\r\n
```

3.3.5 USE_ENDSTOP

Enable USE_ENDSTOP if the homing ensor magnet is properly attached to the focuser in order to be able to do automatic homing.

SET USE_ENDSTOP

Set the use of a homing sensor. It will change the ZEROING behaviour (int 0,1, DEFAULT=0). It should be configured (ask user) on the first initialisation of a new focuser (see ZEROING).

Example command:
```
$BS SET USE_ENDSTOP:1\r\n
```
Example Answer:
```
$BS OK\r\n
```

GET USE_ENDSTOP

Get the USE_ENDSTOP setting (int 0,1).

Example Command:
```
$BS GET USE_ENDSTOP\r\n
```
Example Answer:
```
$BS STATUS USE_ENDSTOP:0\r\n
```

3.3.6 ZEROING

- If USE_ENDSTOP is enabled, the focuser will move DOWN until the homing-sensor is activated (actually it will move downwards 32767 steps (INT16_MIN), in order to prevent the focuser from moving forever in case the sensor is defective). It will then stop (with a deceleration ramp) and subsequently move back to the exact position the sensor was triggered. Then the absolute position is then set to 0.

- If USE_ENDSTOP is disabled, the current position will be set to 0 (equivalent to "SET POSITION:0").

Example Command:
```
$BS ZEROING\r\n
```
Example Answer:
```
$BS OK\r\n
```

3.3.7 POSITION

The current absolute position of the focuser.

SET POSITION

Override the current position (int32).

Example command:
```
$BS SET POS:0\r\n
```
Example Answer:
```
$BS OK\r\n
```

GET POSITION

Get current focuser position (int32).

Example Command:
```
$BS GET POS\r\n
$BS STATUS POS:1234\r\n
```

3.3.8 LIMIT

LIMIT is the maximum absolute position the focuser will travel to (in outwards direction).

SET LIMIT

Set the upper limit for the focuser travel range (int32).

Example command:
```
$BS SET LIMIT:10000\r\n
```
Example Answer:
```
$BS OK\r\n
```

GET LIMIT

Get the upper limit (int32).

Example command:
```
$BS GET LIMIT\r\n
```
Example answer:
```
$BS STATUS LIMIT:10000\r\n
```

3.3.9 FOCUS

One position can be saved locally on the device using the keypad. This is useful for saving your focus position. The position can also be viewed and changed in the **SteelGoII** settings tab.

SET FOCUS

Save a absolute position locally on the controller. It can be loaded/overwritten with the keypad. (int32).

Example command:
```
$BS SET FOCUS:1234\r\n
```
Example Answer:
```
$BS OK\r\n
```

GET FOCUS

Get the saved focus position (int32).

Example command:
```
$BS GET FOCUS\r\n
```
Example answer:
```
$BS STATUS FOCUS:1234\r\n
```

3.3.10 JOGSTEPS

Set the number of steps for the jogging movement mode.

GET JOGSTEPS

Get the current number of steps for the jogging movement. [uint32]

Example command:

```
$BS GET JOGSTEPS\r\n
```
Example answer:
```
$BS STATUS JOGSTEPS:25\r\n
```

SET JOGSTEPS

Set the number of steps for the jogging movement. [uint32]

Example command:
```
$BS SET JOGSTEPS:50\r\n
```
Example answer:
```
$BS OK\r\n
```

3.3.11 SINGLESTEPS

The SINGLESTEPS variable may be used to define another JOGGING interval instead of the default 1 step setting. If used with a telescope of f/6 and above, a single step might be too small to be practical for fine focusing.

GET SINGLESTEPS

Get the current number of steps for the single step movement [uint32] (Default = 1).

Example command:
```
$BS GET SINGLESTEPS\r\n
```
Example answer:
```
$BS STATUS SINGLESTEPS:1\r\n
```

SET SINGLESTEPS

Set the number of steps for the 'single' step movement [uint32]. By default this is 1. If the full resolution is not needed, it might be more convenient to have two jogging modes rather than jogging and single step mode. The valid range is [1, JOGSTEPS]

Example command:
```
$BS SET SINGLESTEPS:10\r\n
```
Example answer:
```
$BS OK\r\n
```

3.3.12 CURRENT_MOVE

Set the digital potentiometer whiper setting controlling the reference voltage VREF, which is inversly proportional to the stepper driver current for movement. $I \propto \frac{1}{V}$
(uint8 [0-127], DEFAULT = 25, safety limit = 10). This is an expert setting and should only be changed if one wants to use a different stepper motor or lift even heavier loads.

SET CURRENT_MOVE

Set CURRENT_MOVE.
Example command:
```
$BS SET CURRENT_MOVE:35\r\n
```

GET CURRENT_MOVE

Get the digital potentiometer whiper setting, controlling the stepper driver current.
(uint8 [0-127])
Example command:

```
$BS GET CURRENT_MOVE\r\n
```

Example answer:

```
$BS STATUS CURRENT_MOVE:35\r\n
```

3.3.13 CURRENT_HOLD

The motor current while in holding position (not moving). The CURRENT_HOLD value is the digital potentiometer whiper setting controlling the reference voltage, which is inversly proportional to the stepper driver current in STOPPED state. $I \propto \frac{1}{V}$ (uint8 [0-127], DEFAULT = 100, Safetly limit=10).

SET CURRENT_HOLD

Set the CURRENT_HOLD variable. Example command:

```
$BS SET CURRENT_HOLD:100\r\n
$BS OK\r\n
```

GET CURRENT_HOLD

Get the digital potentiometer whiper setting, controlling the stepper driver current while the focuser is STOPPED (uint8 [0-127], DEFAULT = 100).
Example command:

```
$BS GET CURRENT_HOLD\r\n
```

Example answer:

```
$BS STATUS CURRENT_HOLD:100\r\n
```

3.3.14 RCX

Set the RCA and RCB digital potentiometer whiper setting, controlling the Bridge A/B blanking and off time of DRV8811 stepper driver (uint8 [0-127]). But beware, setting a too short off-time will overheat the stepper motor! This is an expert setting and should only be changed if using a different stepper motor.

SET RCX

Set RCX variable. Example command:

```
$BS SET RCX:64\r\n
$BS OK\r\n
```

GET RCX

Get the RCX variable. Example command:

```
$BS GET RCX\r\n
```

Example answer:

```
$BS STATUS RCX:64\r\n
```

3.4 GET MOVEMENTS

The controller saves the movements in a rolling circular buffer with size 5.
Format: NB,TIME,DELTA,START,STOP,TEMP0,TEMP1,TEMP_AVG,SOURCE;...

- NB: Sequential numbering of movement sinca last reboot.
- TIMESTAMP: Time [ms] since last movement.
- DELTA: Number of steps moved.
- START: Absolute position at the start of the movement in steps.
- STOP: Absolute position at the end of the movement in steps.
- TEMP0: Temperature [$°C$] of the temperature sensor plugged into the motor unit.
- TEMP1: Temperature [$°C$] of the temperature sensor plugged into the controller unit.
- TEMP_AVG: Average temperature.
- SOURCE: Source of the movement command. [KEYPAD,REMOTE,TCOMP] KEYPAD: A movement was issued by the controller keypad. REMOTE: A movement was issued over the USB connection (eg. by SteelGo II or ASCOM etx.) TCOMP: A movement was issued by the automatic temperature compensation.

```
$BS GET MOVEMENTS\r\n
```

Example answer:

```
$BS STATUS
   MOVEMENTS:6,5393,28889,111,29000,-128.00,23.99,23.99,KEYPAD;\r\n
```

3.5 Temperature compensation related commands

The build in temperature compensation is controlled with three variables. Once the temperature compensation is started, the temperature is saved and continuously compared to the current temperature. If the difference ΔT exceeds the set variable TCOMP_DELTA and the time since the last compensation movement exceeds TCOMP_PERIOD, the focuser will move $\Delta T \cdot$ TCOMP_FACTOR.
The temperature compensation may also be paused (while still keeping track of the temperature change).
The sensor used for the compensation can eb set to either one of the two possible sensors or the average temperature of both.

3.5.1 TCOMP

Enable/Disable the automatic temperature compensation.

GET TCOMP

Get the temperature compensation status (1 == active, 0 == inactive)

```
$BS GET TCOMP\r\n
```

Example answer:

```
$BS STATUS TCOMP:0\r\n
```

SET TCOMP

Set the temperature compensation status (1 == active, 0 == inactive)

```
$BS SET TCOMP:1\r\n
```

Example answer:

```
$BS OK\r\n
```

3.5.2 TCOMP_FACTOR

GET TCOMP_FACTOR

Get the temperature compensation factor $[\frac{steps}{^\circ Celsius}]$.

```
$BS GET TCOMP_FACTOR\r\n
```

Example answer:

```
$BS STATUS TCOMP_FACTOR:0.12\r\n
```

SET TCOMP_FACTOR

Set the temperature compensation factor (float) $[\frac{steps}{^\circ Celsius}]$.

```
$BS SET TCOMP_FACTOR:0.12\r\n
```

Example answer:

```
$BS OK\r\n
```

3.5.3 TCOMP_PERIOD

Do compenation movements only every period.

GET TCOMP_PERIOD

Get the temperature compensation period (uint32_t) $[ms]$.

```
$BS GET TCOMP_PERIOD\r\n
```

Example answer:

```
$BS STATUS TCOMP_FACTOR:30000\r\n
```

SET TCOMP_PERIOD

Set the temperature compensation period (uint32_t) [ms].

```
$BS SET TCOMP_PERIOD:10000\r\n
```

Example answer:

```
$BS OK\r\n
```

3.5.4 TCOMP_DELTA

δT must be exceeded for the temperature compensation movement to be triggered.

GET TCOMP_DELTA

Get the temperature compensation ΔT threshold [$°C$].

```
$BS GET TCOMP_FACTOR\r\n
```

Example answer:

```
$BS STATUS TCOMP_FACTOR:0.5\r\n
```

SET TCOMP_DELTA

Set the temperature compensation ΔT threshold.

```
$BS SET TCOMP_DELTA:0.5\r\n
```

Example answer:

```
$BS OK\r\n
```

3.5.5 TCOMP_PAUSE

Pause the temperature compensation. The temperature drift will be tracked while pause is enabled, once pause is disabled again, a correction movement is issued if needed.

GET TCOMP_PAUSE

Get the temperature compensation pause status (1 == paused, 0 == not paused). The ΔT will be tracked, just the compensation movement is suspended while paused.

```
$BS GET TCOMP_PAUSE\r\n
```

Example answer:

```
$BS STATUS TCOMP_PAUSE:0\r\n
```

SET TCOMP_PAUSE

Set the temperature compensation pause status (1 == paused, 0 == not paused).

```
$BS SET TCOMP_PAUSE:1\r\n
```

Example answer:

```
$BS OK\r\n
```

3.5.6 TCOMP_SENSOR

Sensor used for the temperature compensation function (0 = TEMP0 (motor), 1 = TEMP1 (controller), 2 = average).

GET TCOMP_SENSOR

Get the Sensor used for the temperature compensation.

```
$BS GET TCOMP_SENSOR\r\n
```

Example answer:

```
$BS STATUS TCOMP_SENSOR:0\r\n
```

SET TCOMP_SENSOR

Set the Sensor used for the temperature compensation.

```
$BS SET TCOMP_SENSOR:2\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6 Temperature PWM and PID Control related commands

The 12V/1A PWM output may be used for dew-heaters and in conjunction with one or more temperature sensors, for the precise PID temperature control of instruments.

3.6.1 TEMPX

Temperature of sensor nb. X.

GET TEMP0

Get the current temperature from the temperatur sensor plugged into the stepper motor unit (float). If no sensor is attached (or the sensor is defective), this will read "-128.0".

```
$BS GET TEMP0\r\n
```

Example answer:

```
$BS STATUS TEMP0:17.01\r\n
```

GET TEMP1

Get the current temperature from the sensor plugged into the controller board (float). If no sensor is attached (or the sensor is defective), this will read -128.0 .

```
$BS GET TEMP1\r\n
```

Example answer:

```
$BS STATUS TEMP1:22.56\r\n
```

3.6.2 TEMPX_OFS

TEMP0_OFS and TEMP1_OFS hold a constant offset for the measured temperatures TEMP0 and TEMP1. May be used for precise temperature calibration.

GET TEMP0_OFS

Get the temperature offset for temperatur sensor plugged into the stepper motor unit (float, DEFAULT = 0.0).

```
$BS GET TEMP0_OFS\r\n
```

Example answer:

```
$BS STATUS TEMP0_OFS:0.00\r\n
```

SET TEMP0_OFS

Set a constant temperature offset for temperatur sensor plugged into the stepper motor unit (float, decimal point MUST be given!).

```
$BS SET TEMP0_OFS:-0.43\r\n
```

Example answer:

```
$BS OK\r\n
```

GET TEMP1_OFS

Get the temperature offset for sensor TEMP1 (float, DEFAULT = 0.0).

```
$BS GET TEMP1_OFS\r\n
```

Example answer:

```
$BS STATUS TEMP1_OFS:1.12\r\n
```

SET TEMP1_OFS

Set the temperature offset for the temperature sensor. (float, decimal point MUST be given!)

```
$BS SET TEMP1_OFS:1.00\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.3 PID_CTRL

Enables or disables the PID control. If PID control is enabled and the PWM variable is overwritten, PID_CTRL is automatically disabled.

GET PID_CTRL

Get the status of the PID temperature control. (1 = activated, 0 = deactivated)

```
$BS GET PID_CTRL\r\n
```

Example answer:

```
$BS STATUS PID_CTRL:1\r\n
```

SET PID_CTRL

Enable/Disable the status of the PID temperature control. (1 = activated, 0 = deactivated). If no valid temperature sensor data is obtained, the power output will be disabled.

```
$BS SET PID_CTRL:0\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.4 PID_TARGET

The PID_TARGET is the target temperature (or setpoint) for the PID control. The PID function will vary the PWM power output so that the set PID_SENSOR will converge towards the PID_TARGET.

GET PID_TARGET

Get the current target temperature of the PID temperature control (float).

```
$BS GET PID_TARGET\r\n
```

Example answer:

```
$BS STATUS TEMPTARGET:27.00\r\n
```

SET PID_TARGET

Set the current target temperature of the PID temperature control (float, decimal point MUST be given!).

```
$BS SET PID_TARGET:31.00\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.5 PID_SENSOR

The PID_SENSOR variables holds the sensor number of the temperature sensor to be used as input for the PID control. (0 == Motor, 1 == Controller)

GET PID_SENSOR

Get the current sensor used for the PID temperature control. [0 = motor ,1 = controller ,2 = average]

```
$BS GET PID_SENSOR\r\n
```

Example answer:

```
$BS STATUS PID_SENSOR:0\r\n
```

SET PID_SENSOR

Set the current sensor used for the PID temperature control. [0 = motor, 1 = controller, 2 = average]

```
$BS SET PID_SENSOR:2\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.6 PWM

The PWM variable hold the current PWM power in percent. It is by default set to 50

GET PWM

Get the current PWM value of the 12V/1A PWM power output (uint8 [0,100], 0=OFF, 100=max power).

```
$BS GET PWM\r\n
```

Example answer:

```
$BS STATUS PWM:13\r\n
```

SET PWM

Force the PWM value of the 12V PWM power output (uint8 [0,100], 0=OFF, 100=max power). This will deactivate an active PID temperature controller.

```
$BS SET PWM\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.7 AMBIENT_SENSOR

An AMBIENT_SENSOR may be set (like PID_SENSOR) which will be taken for the automatic dew heater functionality (see AUTO_DEW). [0 = motor sensor, 1 = controller sensor]. (default = 1)

GET AMBIENT_SENSOR

Get the AMBIENT_SENSOR.

```
$BS GET AMBIENT_SENSOR\r\n
```

Example answer:

```
$BS STATUS AMBIENT_SENSOR:0\r\n
```

SET AMBIENT_SENSOR

Set the AMBIENT_SENSOR variable [0,1].

```
$BS SET AMBIENT_SENSOR:1\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.8 PID_DEW_OFS

In PID_DEW_OFS [$\circ Celsius$] a constant offset for the dew heater PID control can be defined. When AUTO_DEW is enabled, the PID controller will adjust the PID_TARGET to the temperature measured by PID_SENSOR + PID_DEW_OFS. That means, the dew heater will be kept at a constant offset with respect to the ambient temperature.

GET PID_DEW_OFS

Get the current PID_DEW_OFS. (float)

```
$BS GET PID_DEW_OFS\r\n
```

Example answer:

```
$BS STATUS: PID_DEW_OFS:4.00\r\n
```

SET PID_DEW_OFS

Set a new PID_DEW_OFS. (float)

```
$BS SET PID_DEW_OFS:5.5\r\n
```

Example answer:

```
$BS OK\r\n
```

3.6.9 AUTO_DEW

When AUTO_DEW is enabled and both, the AMBIENT_SENSOR and the PID_SENSOR read valid temperatures, the PID_TARGET temperature will constantly adjusted to the temperature measured by the AMBIENT_SENSOR + PID_DEW_OFS. This is persitant after reboot.

GET AUTO_DEW

Get the current AUTO_DEW value. (0 = disabled, 1 = enabled)

```
$BS GET AUTO_DEW\r\n
```

Example answer:

```
$BS STATUS AUTO_DEW:0\r\n
```

SET AUTO_DEW

Enable/disable AUTO_DEW. [0,1]

```
$BS SET AUTO_DEW:1\r\n
```

Example answer:

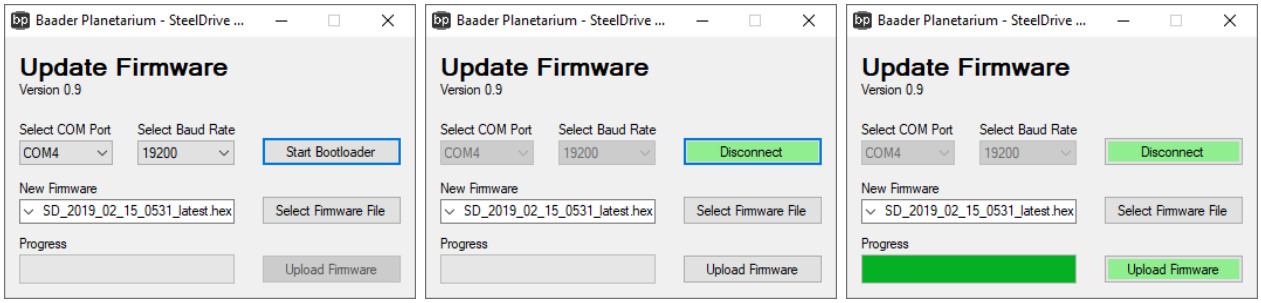
```
$BS OK\r\n
```

Chapter 4

Firmware Update

The windows programm **BP_Steeldrive_II_FirmwareUpdateTool.exe** is provided for updating the firmware of the SteelDriveII controller unit.

- Connect the SDII controller unit to the computer via the provided USB cable.
- Execute **BP_Steeldrive_II_FirmwareUpdateTool.exe**
- The correct COM-Port should already be selected. If not please select the appropriate port.
- Select the new firmware file.
- Click ***Start Bootloader***. If successful the button becomes green.
- now click ***Upload Firmware***. The progress bar visualizes the upload process. Once the firmware is succesfully uploaded, the ***Upload Firmware*** button becomes green.
- Click ***Disconnect***.



Chapter 5

PinOut

5.1 Motor RJ45 connection pinout

The stepper motor unit is connected to the controller unit via a CAT7 LAN cable. This enables the user to easily adapt the cable length to their needs without the need of any special proprietary cable.

NEVER connect either the stepper motor unit or the controller unit to a LAN network via the RJ45 port. It will cause damage to the switch and/or other attached devices!

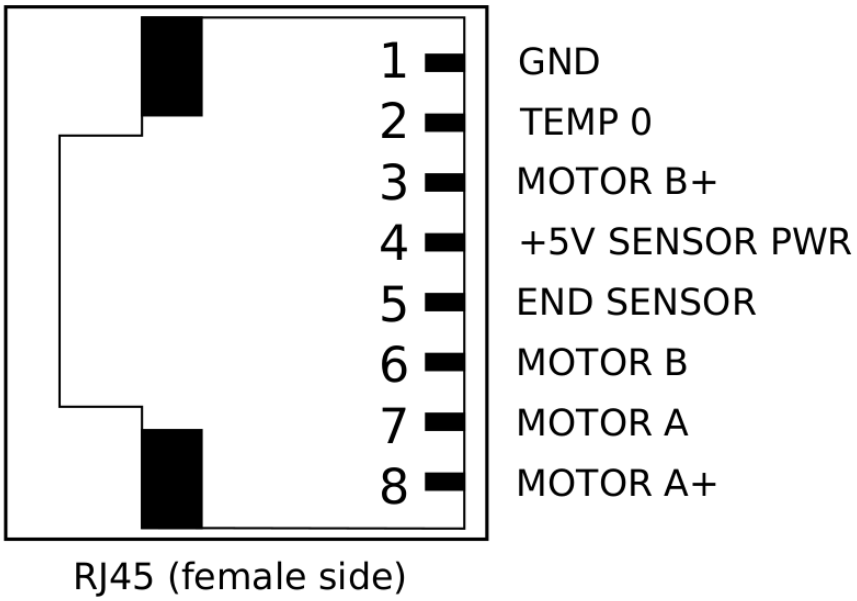


Figure 5.1: RJ45 pinout (female side).

5.2 Mini DIN8 pinout

The Mini-DIN 8 connector of the controller unit provides a 12V/1A PWM power output for dew heaters (Cnich RCA adapter cable), temperature control of other instruments etc. It also provides connection with other microprocessors via SPI.

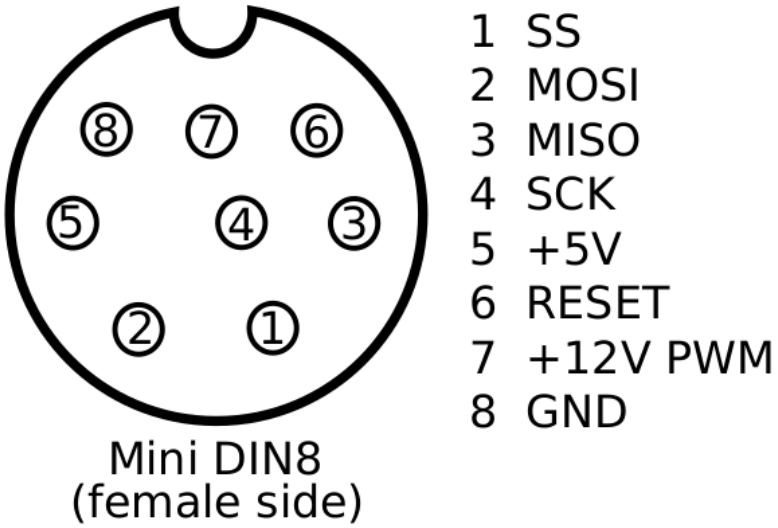


Figure 5.2: Mini-DIN 8 pinout (female side)

5.3 3.5mm jack pinout

The 3.5mm (audio) jacks on both, the motor and the controller unit are used for the attachment of the digital temperature sensors. Multiple sensors (up to 8) may be connected in parallel (eg. via a 3-pole Y-audio adapter).

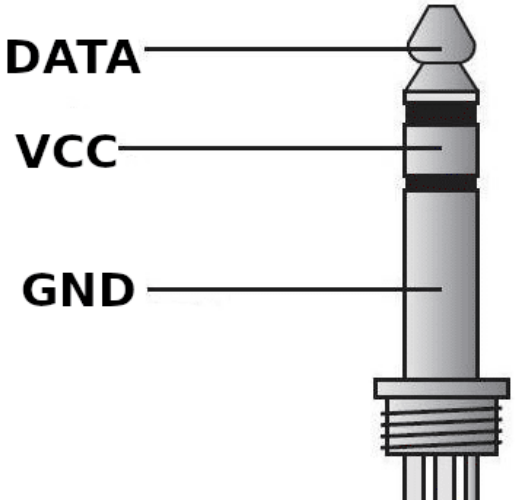


Figure 5.3: 3.5 mm audio jack temperature sensor connector

.1 CRC8 checksum calculation

The Dallas/Maxim CRC8 checksum calculation algorithm is used.

```
/** CRC lookup table */
const uint8_t crc_array[256] = {
        0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83,
        0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
        0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e,
        0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc,
        0x23, 0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0,
        0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62,
        0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d,
        0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff,
        0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5,
        0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07,
        0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58,
        0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a,
        0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6,
        0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24,
        0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b,
        0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
        0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f,
        0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd,
        0x11, 0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92,
        0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50,
        0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c,
        0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee,
        0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1,
        0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73,
        0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49,
        0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b,
        0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4,
        0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16,
        0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a,
        0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
        0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7,
        0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,
};
/** \fn CRC8 (Maxim/Dallas) calculation
 *  \brief Calculate CRC8 checksum based on lookup table
 *  \param data pointer to data array
 *  \param size size of data array
 *  \returns CRC8 checksum
uint8_t crc8(uint8_t * data, uint8_t size)
{
        uint8_t crc = 0;
        for ( uint8_t i = 0; i < size; ++i )
        {
                crc = crc_array[data[i] ^ crc];
        }
        return crc;
}
```