PIO OS

Reference Manual

Table of Contents

1 Introduction

This manual describes the usage of PROTON PIO devices running the PIO Operating System (short: PIO OS). The PIO works as an IP to RS485 bridge and is used to control PROTON cameras from a PC / Mac or HW controller (RCP) via IP.

Details on the general operation can be found in chapter 2. Instructions for firmware updates in chapter 3.

PIO devices and attached cameras can be controlled with PROTON Control, an easy-to-use PC and Mac application (see chapter 4.1). If you want to control the device via a terminal application or a custom hardware controller, see chapter 4.2 for tool recommendations and chapters 5 and following for a detailed description of the command protocol.

Note: This manual only covers “smart” PIO devices with IP support. The original PIO is a simple USB to serial bridge which does not require any internal software and is not covered by this manual.

1.1 Supported Devices

This manual covers the following PIO devices:

- PROTON PIO-E

For a full list of all supported devices and instructions on how to identify them see the `system info` command.

1.2 RCP Compatibility

The following RCP suppliers have integrated support for PIO-E into their HW controllers:

- CyanView: https://www.cyanview.com/

2 General Operation

A PIO has three connectors:

1. Power and control: RJ45 ethernet port with PoE 802.3af.
2. Camera: 6 pin Hirose HR10 connector for power and RS485 control.
3. Service: USB-C port for local serial connection.

The interfaces are described in detail in the following chapters.

The device will immediately turn on when the ethernet port is connected to a compatible PoE switch or power injector, the boot process takes a few seconds. Once the device is operational the LED on the back side will blink purple or green. For details on the status LED see chapter 2.4.

The PIO itself and its attached camera device are controlled either remotely via an IP connection or locally via the USB port. IP control is possible via PROTON Control (see chapter 4.1) or supported RCPs (see chapter 1.2). Control via the USB port is only possible with PROTON Control or via a terminal application.

PIO OS supports up to 8 concurrent connections via IP, see chapter 7 for details.

The following diagram shows the connection plan for the full system consisting of one or more controllers, a PoE switch, the PIO-E and one or more camera devices. Note that using more than one camera requires a special Y breakout cable and the maximum power output of 12 W must not be exceeded.

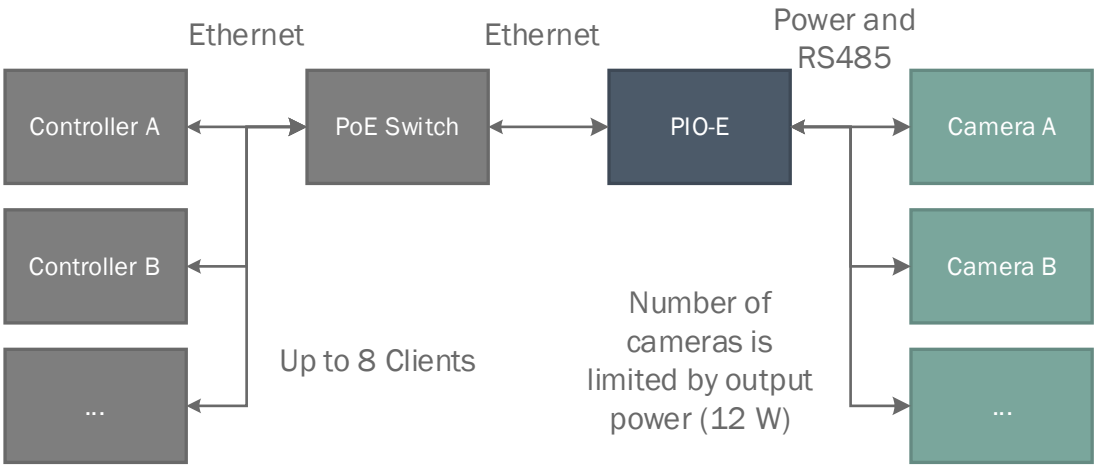

Figure 1: PIO-E connection plan.

2.1 Ethernet Port with PoE

The PIO-E is powered and controlled via a RJ45 connector for ethernet with PoE. It is compatible with PoE switches and power injectors which comply to the IEEE 802.3af PoE standard and supply at least 40 V (typically 48 V). The maximum power drawn from the switch is less than 15 W depending on the attached device(s). When connecting multiple PIOs to the same switch, make sure to not exceed the switch's maximum power output.

The ethernet port supports speeds of 10 / 100 Mbit/s.

By default, the ethernet port is configured with the following IP address:

```
IP:       10.0.0.101
Netmask:  255.255.255.0
Gateway:  10.0.0.1
```

DHCP is enabled by default so the above settings will only be used if no DHCP server is available.

Note: PIO devices only support IPv4!

For details on how to find PIO devices in the network and control them via the WebSocket protocol see chapter 7.

2.2 Camera Connector

The PIO provides power and RS485 control to a camera via a 6 pin Hirose HR10 connector. The pin assignment is identical to the connector on PROTON cameras:

Table 1: Pinning of the power and control connector.

Pin	Function
1	RS485 A / +
2	RS485 B / -
3	Unused
4	Unused
5	Ground
6	Supply Voltage (15 V)

2.2.1 RS485 Baud Rate

The RS485 output of the PIO supports all typical baud rates which may be used with PROTON camera devices: 250000, 230400, 115200. 57600, 19200, 14400 and 9600 baud. To change the baud rate of the RS485 interface use the `system baudrate` command (see chapter 9.3.10).

2.2.2 Camera Power

PIO-E always supplies 15 V to the attached device(s). The maximum power it can deliver is 12 W, which makes it **compatible with all PROTON cameras except the HFR** which draws up to 15 W.

The power to the attached device(s) can be turned on or off with the `system power` command (see chapter 9.3.13 for details). This can be used to power cycle the devices or turn them off permanently to conserve power (e.g. in battery powered scenarios). Power can be toggled via PROTON Control, see chapter 4.1.2 for details.

2.3 USB Service Port

In addition to IP control the PIO also allows local control via the USB-C port. Connecting the port to a PC or Mac creates a virtual serial port device which can be opened with PROTON Control or the usual terminal tools (see chapter 4 for details).

The service port is intended for the following use cases:

- Update the PIO device or attached devices without an IP connection (see chapter 3).
- Local camera setup in the field with PROTON Control or a terminal application (see chapter 4).
- Debugging and analyzing: All data which is sent to the connected cameras via the ethernet port is also mirrored to the service port so it can be used as a data logger to debug issues in the field.

Since this is a virtual serial port the selected baud rate of the port does not have any effect on the transfer speed and is ignored by the device. Also, the baud rate of the serial port has no effect on the RS485 interface, you must select the correct baud rate for the attached camera with the `system baudrate` command.

Note: The PIO is **not powered via the Service port**, power must always be supplied via PoE 802.3af to the Ethernet port!

For details on control via the USB service port see chapter 8.

2.3.1 Avoiding high COM Port Numbers on Windows

In contrast to Linux the Windows operating system assigns a new COM port number to each device that is connected to the PC. By default, Windows determines if a device is new by looking at its VID, PID and HW serial number. Since each PIO has a unique HW serial number, Windows treats each one as a new device and assigns a new COM port number to it.

If your application requires it, you can tell Windows to ignore the hardware serial number for specific devices which allows it to reuse the COM port for multiple PIO devices (if only one device is connected at a time).

To enable this feature, use the Windows Registry Editor to create the following **binary** registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbflags\IgnoreHWSerNum303A1001
```

And set its value to **01**.

This tells Windows to ignore the HW serial number for devices with a VID of 0x303A and a PID of 0x1001 (the IDs of the USB chip in the PIO, can be read via the Windows Device Manager).

Note: This is not required for usage of the PROTON Control and Updater applications as they automatically scan all available COM ports and do not require fixed port numbers.

2.4 Status LED

On the top of the device is an RGB status LED that indicates the current device state. The following blink codes are possible:

Table 2: Status LED blink codes.

State	Blink Code	Description
Boot	**Static cyan**	Device is booting after power got connected. This state is very short so normally this code is barely visible for a few milliseconds. If a firmware installation is interrupted due to power-loss this state will take longer while the bootloader recovers.
Verify / Upgrade Firmware	**Static green**	Device is verifying or installing a firmware. This happens both during normal boot (a few seconds) and during a firmware upgrade (about two minutes).
Boot Error	**Static red**	Device failed to start. This is a critical error that cannot be recovered. Contact PROTON customer support.
Static IP	**Blinks purple**, 2.5x per second (200ms on, 200ms off)	Device uses a static IP address, either because DHCP is disabled or the DHCP server has not assigned an IP address yet (or is unreachable).
DHCP	**Blinks green**, 2.5x per second (200ms on, 200ms off)	Device has received an IP address via DHCP.
Connected	**Blinks blue**, 2.5x per second (200ms on, 200ms off)	Device is connected to a PROTON Control application or HW controller (RCP).
Error	**Blinks red**, 2.5x per second (200ms on, 200ms off)	Device encountered an error. This should not happen during normal operation. To get the error log use the `system error` command. If the error persists, contact PROTON customer support.
Button Pressed	**Static red**	The LED turns red as feedback for the push button being pressed.
Reserved	**Blinks red / white**, 2.5x per second (200ms red, 200ms white)	The button has been pressed for more than 2s. This is reserved for future use.
Reset	**Static red**	The button has been pressed for more than 8 seconds. Release the button to perform a reset to default settings.

2.5 Push Button

The PIO has a push button on the top next to the status LED. If the button is pushed this is indicated by the LED switching to static red color. Depending on how long the button is pressed different actions are triggered as described in the following chapters.

2.5.1 Reserved Button Action (2s)

After holding the button for 2s the LED will switch from static red to toggling between red and white. This action is currently reserved for future use, releasing the button now has no effect and the PIO resumes normal operation.

2.5.2 Settings Reset (8s)

When holding the button for 8s the LED will switch back to solid red. Releasing the button now will perform a settings reset (see chapter 6.1 for details) and reboot the device.

This enables DHCP again and restores the default static IP address configuration (see 2.1) and can be used to make the device reachable again in case its IP configuration is invalid.

2.6 Error Recovery

In case the device runs into an unexpected condition several reporting and recovery mechanisms are implemented.

Should the PIO lock up due to faulty firmware, instable power-supply or other unexpected errors the internal watchdog will reboot the device trying to resume operation. When this happens the status LED blinks red afterwards and the `system error` command shows that a watchdog event was logged:

```
→ 101 system error
← Watchdog: System got reset by watchdog.
← OK
```

The error flags are cleared when the PIO starts successfully after a power-cycle or reboot.

If the system keeps restarting due to watchdog events, try to reset the device to default settings or perform a firmware update as described below.

2.6.1 Recovery via Settings Reset

To recover a PIO device first try to reset the settings to their default values (see chapter 6.1). The easiest way to perform a reset is by holding the push button for 8 seconds, see chapter 2.5.

2.6.2 Recovery via Firmware Update

If the PIO is still showing watchdog errors after the reset, try to perform a firmware update as described in chapter 3.

3 Firmware Update

Note: This chapter describes updates of the PIO itself. Camera devices connected to the PIO are updated as usual since the PIO works like a transparent bridge to the PROTON Control and PROTON Updater (USB only) applications.

Firmware updates are performed via the PIO's IP or USB interfaces. The update is a two-step process:

1. Transfer new firmware to the device.
2. Install and verify new firmware.

This process is executed automatically by the PROTON Control (IP and USB) or PROTON Updater (USB only) software which are described in the following chapters. For details on the update process see chapter 3.3.

When the device is switched into firmware update mode it gracefully disconnects all WebSocket clients except the one who requested the update and stops relaying commands to attached camera devices. This ensures that the update process does not get interrupted. New WebSocket connections are only accepted after the update is finished.

To avoid the device getting stuck in firmware update mode the device automatically reboots and discards any uploaded update data if the client does not provide new update data for 1 minute.

For details on firmware update via WebSocket see chapter 7.3.5, for details on update via USB see chapter 8.2.

3.1 PROTON Control (IP and USB)

Firmware updates via PROTON Control are described in the Control Software chapter, see section 4.1.

3.2 PROTON Updater (USB only)

Alternatively, firmware updates can also be done via the PROTON Updater application. The app is supplied with every firmware release and runs under Windows and macOS. It can be downloaded here: https://proton-camera.com/downloads/

Note: The updater only works when the PIO is connected via the USB service port. If you want to update the device via IP, use the fully featured PROTON Control application instead (see chapter 4.1).

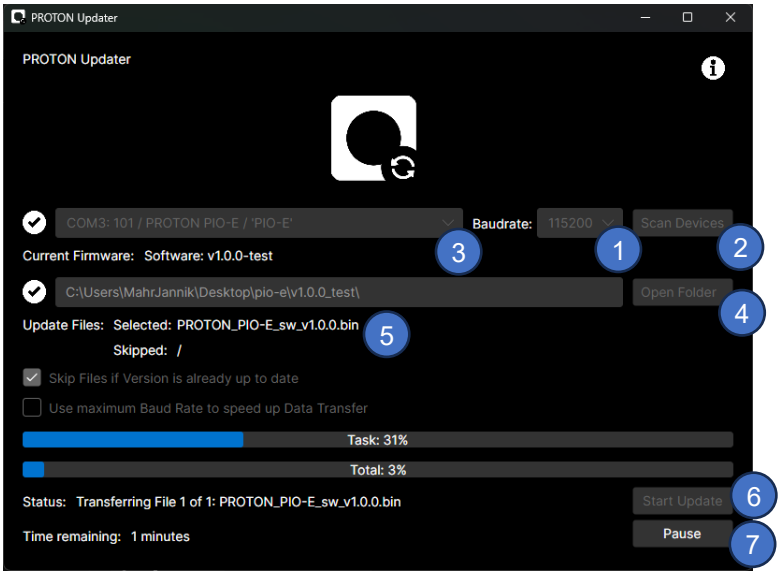

Figure 2: Example for the PROTON Updater application

To install a firmware update with the PROTON Updater GUI, follow these steps:

- Connect the PIO to your PC or Mac using a USB-C cable.
- Open the “PROTON Updater” executable from the firmware release folder.
- The selected baud rate **(1)** is “don’t care” for PIO devices so leave it as is and click *Scan Devices* **(2)**, the GUI will automatically scan all available serial ports ports for connected PROTON devices and show them in the drop-down menu besides the *Scan Devices* button.
- If only one device is connected it will automatically be selected. If you have multiple devices connected select the PIO which you want to update from the drop-down list **(3)**.
- Click the *Open Folder* button **(4)** and navigate to the folder which contains the firmware updates you want to install.
- Verify that the GUI has found the correct update files, they are displayed below the update folder path **(5)**.
- Click *Start Update* **(6)** The whole process takes about 2 minutes. The upper progress bar will fill several times (once for each update file and finally for the install and verify step). The overall progress and the estimated total time remaining are displayed at the bottom of the window.
- The process can be aborted anytime by clicking the *Abort* button **(7)**. Unless you close the GUI or disconnect the PIO, the progress is retained, and you can continue by clicking the *Start Update* button **(6)** again.
- **Note:** Only after all files have been transferred successfully, is the update made permanent. If the PIO is power cycled before all files are transferred, the progress is lost.

3.3 Update Process

This chapter is provided for reference only; users can use the PROTON Updater or PROTON Control for firmware updates which automatically perform the required steps.

3.3.1 Data Transfer

Before data transfer starts, the PIO must be switched to firmware update mode, see `system update` command for details. The update process is robust regarding interruptions and data corruption.

In update mode all other WebSocket connections are disconnected and new connections are rejected. If the client does not supply update data for 1 minute the device reboots automatically to abort the update and become reachable again.

3.3.2 Install and Verify

After the firmware has been transferred successfully the PIO is restarted with the `system reboot` command. During boot it will detect the new firmware, install and verify it. This process takes about **30 seconds**. During installation the status LED will glow static cyan and green. In case the image cannot be verified by the bootloader (wrong image uploaded, data got corrupted) the update will be aborted, and the PIO starts with the previous firmware.

After the firmware has been installed and verified the PIO continues to boot. During the initialization of the application a self-check is performed. Should the PIO not be able to initialize, the status LED will blink red. In this case the PIO will revert to the previous firmware when it is power cycled or rebooted with the `system reboot` command.

4 Control Software

4.1 PROTON Control (IP and USB)

All features of the PIO and attached PROTON cameras can be controlled via the PROTON Control application which runs under Windows and macOS. The app is supplied with every firmware release and can also be downloaded here: https://proton-camera.com/downloads/

While PROTON Control is mainly used to control the camera devices which are attached to the PIO, it can also be used to set up the PIO itself. It works both locally via USB and remotely via IP.

4.1.1 Camera Control via PIO

4.1.1.1 Automatic Discovery

By default, PROTON Control is set to “Scan All” on both serial and network interfaces:

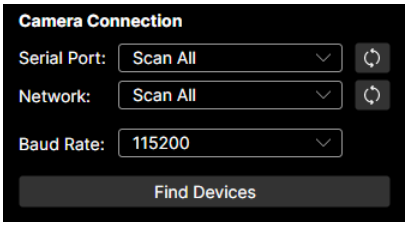

Figure 3: PROTON Control: Automatic device detection

Clicking the “Find Devices” button will list all cameras which are attached to a PIO that is connected via USB or can be found in the network via mDNS (see section 7.2).

4.1.1.2 Manual IP

If automatic discovery does not work in your network, the IP address of the PIO can be entered manually by selecting “Manual Host/IP” in the “Network” drop-down menu as shown below:

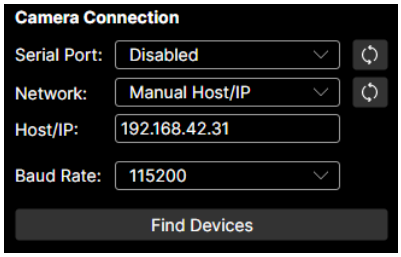

Figure 4: PROTON Control: Manual IP selection

If you know there are no cameras connected via USB / serial port you can also set the “Serial Port” drop-down menu to “Disabled” to further speed up the connection process.

4.1.1.3 Camera Control

Once connected, all camera controls of PROTON Control work as usual. PROTON Control lists the device prefixed with the user defined name of the PIO (here “PIO-E”) instead of the serial port:

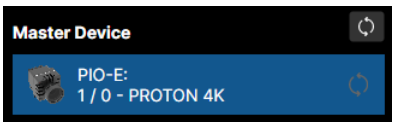

Figure 5: PROTON Control: Camera connected via PIO-E

4.1.2 Configure and Update PIO Devices

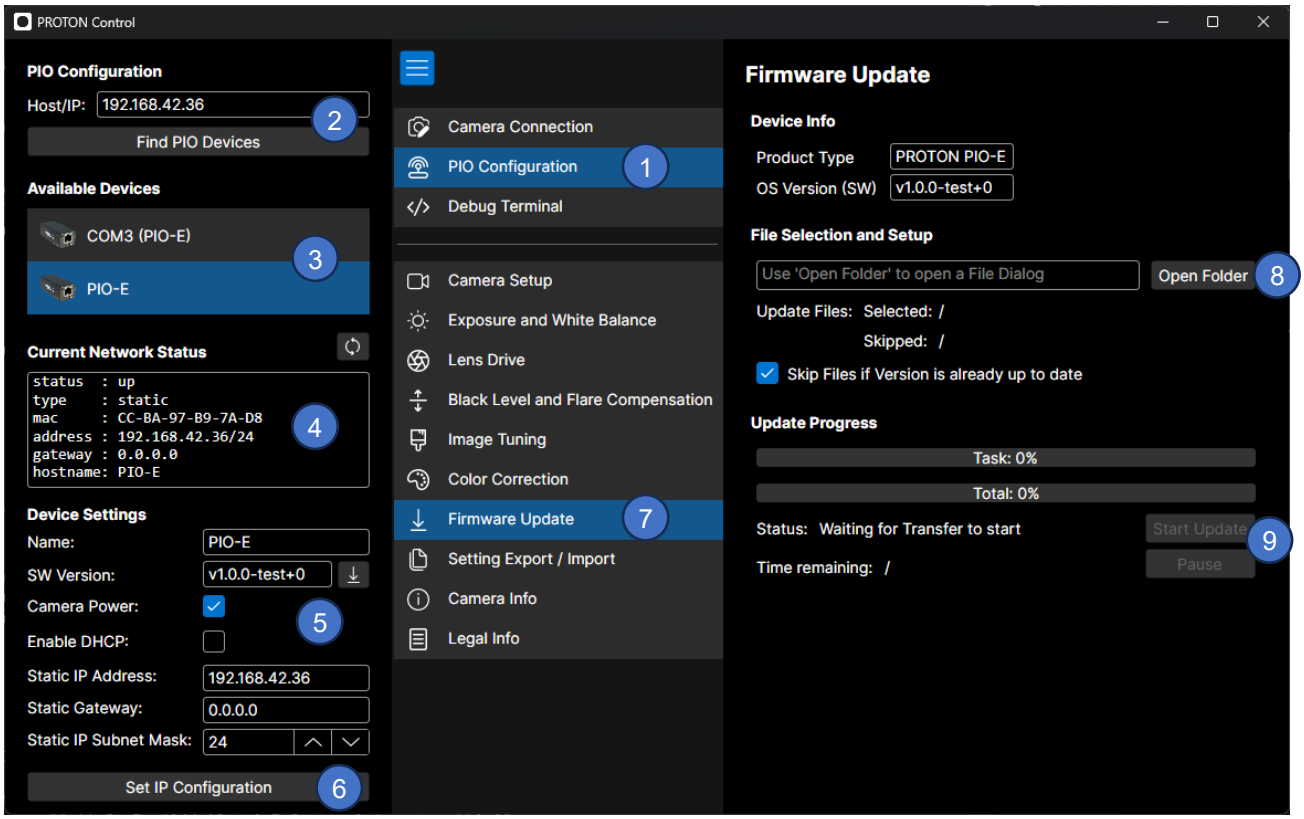

Figure 6: Configure PIO via PROTON Control

To configure a PIO device (e.g. change its IP address), connect it via USB or make sure it is reachable on the network. Then open PROTON Control and:

- Select the “PIO Configuration” tab **(1)** from the selection pane. Now click “Find PIO Devices” **(2)** to search for PIO devices on both IP and local USB connections. If automatic device detection via mDNS (see section 7.2) does not work in your network, you can also manually enter the IP address of the PIO.
- Select the PIO device which you want to configure **(3)**. Note that devices which are connected via USB are listed as a serial port (COM3 in this case) with the PIO name in braces.
- The current network status of the PIO is listed in **(4)** and can be updated by pressing the refresh button.
- You can adjust the PIO’s name by simply typing in the name field, it is updated immediately. The network settings **(5)** are only applied after pressing the “Set IP Configuration” button **(6)** so that all network related changes are applied together.
- You can also enable or disable power to the attached devices **(5)**. Note that this setting is permanent, so if power is turned off it will stay off even after a power cycle until you turn it on again. You will not be able to find devices attached to the PIO-E if power is turned off.

- To update the PIO itself click on the small update button besides the firmware version or simply open the update tab **(7)** while a PIO device is selected. In the update tab select the folder which contains the firmware image **(8)** and press “Start Update” **(9)**.

4.2 Terminal Applications (USB only)

For direct control via commands on the USB service port, you can also use any terminal application which supports opening serial ports including:

- Putty: https://www.putty.org/
- Tera Term: https://teratermproject.github.io/index-en.html
- Serial Monitor for VS Code: https://marketplace.visualstudio.com/items?itemName=ms-vscode.vscode-serial-monitor

5 Command Interface

PIOs and the attached camera devices are controlled via a text-based command interface on both the IP and USB connection. All commands consist of human readable ASCII characters. A command consists of the device ID followed by one or multiple command words and no, one or multiple parameters:

```
<ID> <command name> <parameters>
```

Both the PIO and the attached camera store each received character in an input buffer until a CRLF (Carriage Return and Line Feed or “\r\n”) is received. Then the command is evaluated and, if the address matches one of the device’s addresses, executed.

The PIO has a device ID of `101` to be outside of the camera address range which goes from `0` to `100`. Since there can never be more than one PIO device on an IP or USB connection the ID is fixed. This allows a controller to treat the PIO like any other PROTON device. From a functional point of view the PIO behaves like an additional device on a virtual RS485 bus:

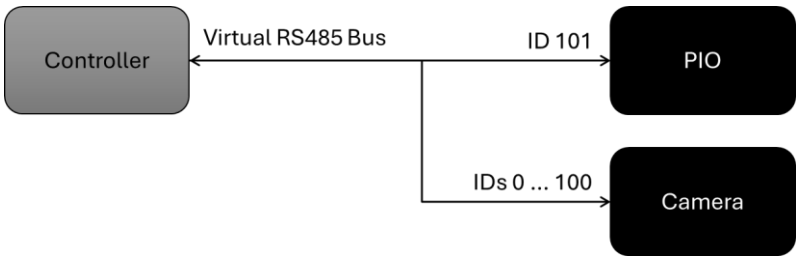

Figure 7: Virtual RS485 Bus

For example, to set the name of a PIO device the following command is used:

```
→ 101 system name My PIO Name
← OK
```

The PIO replies to commands with `OK` or `FAIL <error code>` just like other PROTON devices. During command execution no new commands should be sent until the device responds. Since this is a PIO command, it is not visible on the RS485 interface.

A typical command to change the video mode of a camera would be:

```
→ 1 video mode 12
← OK
```

Where the camera has ID `1`. Since this is not a PIO command, it is forwarded to the RS485 interface.

In summary there are two types of commands:

1. **Camera Commands** on **addresses `0` to `100`** are ignored by the PIO and forwarded to the attached camera device(s) instead.
2. **PIO Commands** on **address `101`** are processed by the PIO itself.

Note: The baud rate of the RS485 interface must be adjusted to match the configured baud rate of the camera. This is done with the PIO command `system baudrate`. By default, it is set to 115200 baud which is also the default on PROTON cameras. Also see chapters 2.2.1 and 9.3.10.

5.1 Connection Types

The **IP connection** is done via the WebSocket protocol using plain text data. See chapter 7 for details and an example implementation of a WebSocket client.

The **USB service interface** opens a virtual serial port (see chapter 8 for details).

The PIO works as a bridge between the controller and the camera:

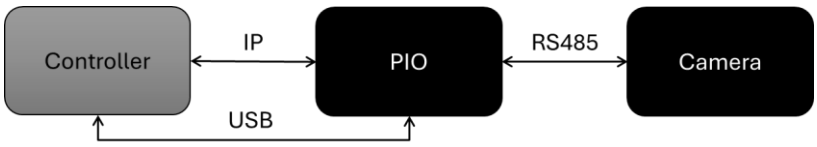

Figure 8: Device Communication Schematic

The PIO forwards all camera commands (commands which are not received on the PIO's device address `101`) from the controller to the camera and vice versa.

5.2 Timeout Handling

To avoid commands from multiple clients or different interfaces (USB / IP) getting mixed up the PIO only allows one camera command to be executed at a time. Should a device not reply to a command because it was malformed or the device got disconnected a timeout is registered and the command returns with error code -116.

The timeout is configured independently for each interface and in case of WebSocket connections also independently for each client using the `system timeout` command. It is the responsibility of the controller to set a sufficiently long timeout for the command. For maximum command runtimes see the reference manual of the connected device.

For details on timeout handling see chapters 7.3.2 and 8.1.

5.3 Parameter Data Types

The following parameter data types are supported:

- **Signed Decimals**, e.g.: `-2947`, `40687`
- **Signed Hexadecimals**, e.g.: `-0x100`, `0x123AF7`
- **Strings**, e.g.: `plain_text_string`
- **Booleans**: For commands that accept a Boolean parameter, like an enable flag, the following values can be used:
  - `0` or `1`
  - `true` or `false`
  - `on` or `off`
  - `enable` or `disable`

  For an example the following commands have the same effect:

```
system power 1
system power true
system power on
system power enable
```


5.4 Hierarchical Command Structure

Commands are structured hierarchically, that means a command can have subcommands. A command string is built by concatenating command words starting from the top level. For an example the command to list supported RS485 baud rates is:

```
101 system baudrate list
```

It is part of the `system` top-level command group which has the `baudrate` subcommand group which provides the `list` command.

5.5 Command Types

The following chapter lists the different command types.

5.5.1 Direct Commands

Commands that do not change a setting but execute a fixed function are called “Direct” commands, e.g. the `system update` command.

5.5.2 Setter and Getter Commands

Most of the commands provide two modes, a “Setter” mode to change a setting and a “Getter” mode to retrieve the current value of the setting.

A setter commands takes one or multiple parameters and applies the given values. The reply only consists of either `OK` or `FAIL <error_code>` and no further output.

Example:

```
→ 101 system baudrate 57600
← OK
```

A getter command takes no arguments and replies with the command name followed by one or multiple values and is terminated by either `OK` or `FAIL <error_code>`.

The above `system baudrate` command can be called without arguments to act like a getter command:

```
→ 101 system baudrate
← system baudrate 115200
← OK
```

5.5.3 Pure Getter Commands

There are also commands which are pure getters, that means they do not have a setter function, e.g.:

```
→ 1 system runtime
← system runtime 237700
← OK
```

5.5.4 Getter Commands with Arguments

These are special getter commands which require one or multiple arguments.

5.5.5 List Commands

Some commands have a `list` subcommand (e.g. `system baudrate list`) which lists all valid options for this command. Example:

```
→ 101 system baudrate list
← 9600
← 14400
← 19200
← 57600
← 115200
← 230400
← 250000
```

← OK

5.5.6 Special Commands

Some commands combine multiple of the above modes or are completely unique, for an example the `system info` or `eth ipconfig` commands will only print information but do not have the leading command name.

These intricacies are described in detail in chapter 8.2.1.1.

5.6 Error Codes

The following table lists the error codes which can be returned after the `FAIL` keyword:

Table 3: Common command error codes.

Error Code	Description
1	The help message was printed because the command was malformed.
-8	Command not found: The command is unknown and cannot be executed.
-14	An error occurred during command execution.
-16	Command cannot be queued because command queue is full, command is dropped.
-19	Missing device: An internal device is missing or does not respond.
-22 or -34	Invalid parameter value(s): The given parameters are outside the valid value range.
-28	The given parameter is too long (e.g. for the device name).
-71	Invalid number of parameters: The number of parameters is not supported by the command.
-96	Invalid command received, command is dropped.
-111	Synchronization to master device failed: Invalid master signal (e.g. video mode mismatch).
-113	Synchronization to master device failed: No master signal detected (loss-of-link).
-116	Command Timeout: Connected device did not reply within the allowed timeout (see `system timeout` command).
-128	Device is not connected to a client.
-134	Operation not supported: The requested operation is not supported by this device.
-140	Operation is currently not allowed because another setting blocks it or the device is in firmware update mode.

For an example the `system baudrate` command expects exactly 1 parameter. If it is called with 2 parameters, the device replies with error -71:

```
→ 101 system baudrate 115200 1
← FAIL -71
```

5.7 Command Alias

To support shortened command names, a command can have an alias. Instead of using the full command syntax, the alias can be used. For an example the command `system baudrate` can also be called by its alias `baudrate`.

To get a list of all available aliases use the `alias` command. Example:

```
→ 101 alias
← Available command alias:
←   ipconfig -> eth ipconfig
←   ip -> eth ip
←   gw -> eth gateway
←   ...
← OK
```


Aliases can also be combined with the normal command syntax. For example, to list all RS485 baud rates you can use any of:

`system baudrate list` (full syntax)
`baudrate list` (`baudrate` alias for `system baudrate`)

5.8 Built-in Help

PIO OS includes extensive help messages for all commands. To list general help instructions and a list of all top-level commands, use the `help` command.

To get specific help messages for a command send the command name followed by `-h`. Example:

```
→ 101 system baudrate -h
← baudrate - [alias: baudrate] [getter]
←            Set RS485 baud rate. This baud rate is used to communicate with
←            devices attached to the RS485 port and it must match the baud rate
←            of the connected devices.
←            To get supported rates use the 'baudrate list' subcommand.
←            Usage: baudrate <rate>
← Subcommands:
←   list  : List all supported baud rates of the RS485 interface.
←           Usage: list
← OK
```

The first line of the reply contains some general information about the command (e.g., does it have an alias). It is followed by a detailed description of the command and its usage. Finally, all subcommands are listed (if it has any).

5.9 Auto Completion

It is possible to call commands without using their full name if the name is unique. For an example the `system baudrate` command could also be called as:

```
→ 101 sys baud
← sys baud 115200
← OK
```

But it is not possible to call the `system runtime` command like this:

```
→ 101 system r
← FAIL -8
```

Because that would be ambiguous with the `system reboot` command.

Note: For getter commands the device always replies with the same command name that was used to query the value, e.g. `sys baud` instead of `system baudrate` for the above example.

Auto completion also works for aliases, for an example you can use `reset` instead of the `reset_settings` alias.

Warning: When new commands are added to the device in the future old commands may become ambiguous. Therefore, it is not recommended to use shortened commands in your controller software!

6 Settings Handling

In contrast to PROTON camera devices which provide a lot of settings which change frequently, PIO devices only have a handful of settings which are mostly static. Therefore, all settings get saved automatically when they get changed and there are no “save” or “load” commands.

6.1 Reset

The device can be reset to its default state using the `settings reset` command. When called without parameters it will only reset non-critical settings.

Use the reset command to make the device accessible again if its network configuration is unknown or invalid. After the reset the device has DHCP enabled, and its static IP is set to 10.0.0.101/24.

To perform a full reset, use the `settings reset all` command, it will also reset the following critical settings:

- Device name (section 9.3.2)

7 IP Control

PIO OS provides a standard TCP/IP v4 stack and the device is reachable via its IPv4 address or hostname. The IP address can be assigned statically or dynamically via DHCP. To automatically detect PIO devices on the network PIO OS implements an mDNS responder.

For control PIO OS allows up to 8 clients to be connected at the same time via WebSocket connections. This allows you to use multiple controllers for different tasks, e.g. dedicated controllers for shading, exposure and lens control.

These functions are described in detail in the following chapters.

7.1 Setting an IP Address

Since PIO-E has no integrated buttons (despite the reset button) or display it usually gets its IP address via DHCP. You can look up the IP address in your DHCP server or find the device using mDNS as described in chapter 7.2.

To switch to a static IP address, you can use PROTON Control (see chapter 4.1.2). If you do not have a DHCP server and cannot reach the PIO device via network a connection via the USB service port is needed to change the network settings (see chapter 8).

If the network configuration of the device is unknown, you can always perform a settings reset by pressing the reset button for 8 seconds which configures the device for DHCP again (see chapter 2.5.2 for details).

7.2 Finding PIO Devices via mDNS

PIO devices respond to mDNS queries on the standard UDP port 5353 and report their capabilities. Any Zeroconf library that supports mDNS discovery should be able to find PIO devices on the network.

In the mDNS response the PIO adds some of its configuration data as properties. This allows the client to quickly identify the PIO and its current setup without having to send commands via WebSocket. The following properties are included:

- `name`: The device name, e.g. “PIO-E”.
- `baudrates`: A comma-separated list of all supported RS485 baud rates, e.g. “9600,14400,19200,57600,115200,230400,250000”.
- `serial`: The device’s serial number, e.g. “CC-BA-97-B9-89-3C”. The serial number is also the device’s MAC address and is lasered to the bottom of the device.
- `version`: The PIO OS version running on the device, e.g. “v1.0.0”.
- `power`: A flag indicating if camera power is turned on, can be “0” or “1”.
- `update`: A flag indicating if the camera is currently in firmware update mode, can be “0” or “1”.

7.2.1 Software Libraries

Popular Zeroconf implementations are Avahi (Linux and macOS) and Bonjour (Apple). The following open-source implementations are recommended:

- C++: https://github.com/HBPVIS/servus

- C#: https://github.com/novotnyllc/Zeroconf
- Python: https://github.com/python-zeroconf/python-zeroconf

7.2.2 Python Example

The following Python script shows how to search for PIO devices and get their properties. It requires the "zeroconf" package which can be installed with pip:

```
pip install zeroconf
```

```python
#!/usr/bin/env python

"""Scan for PROTON PIO devices on all network interfaces via mDNS."""

from zeroconf import ServiceBrowser, ServiceListener, Zeroconf, IPVersion
import pprint

class MyListener(ServiceListener):

    def update_service(self, zc: Zeroconf, type_: str, name: str) -> None:
        if (name.startswith("proton-pio")):
            print(f"Service {name} updated")

    def remove_service(self, zc: Zeroconf, type_: str, name: str) -> None:
        if (name.startswith("proton-pio")):
            print(f"Service {name} removed")

    def add_service(self, zc: Zeroconf, type_: str, name: str) -> None:
        if (name.startswith("proton-pio")):
            info = zc.get_service_info(type_, name)
            ip = info.ip_addresses_by_version(IPVersion.V4Only)[0]
            print(f"Service {name} found at IP {ip}, with the properties:")
            pprint.pp(info.properties)

zeroconf = Zeroconf()
listener = MyListener()
browser = ServiceBrowser(zeroconf, "_http._tcp.local.", listener, )
try:
    input("Press enter to exit...\n\n")
finally:
    zeroconf.close()
listener = MyListener()
browser = ServiceBrowser(zeroconf, "_http._tcp.local.", listener, )
try:
    input("Press enter to exit...\n\n")
finally:
    zeroconf.close()
```

It produces the following output with one PIO device in the network:

```
Press enter to exit...
```

```
Service proton-pio._http._tcp.local. found at IP 192.168.42.33, with the
properties:
{b'name': b'PIO-E',
 b'baudrates': b'9600,14400,19200,57600,115200,230400,250000',
 b'serial': b'CC-BA-97-B9-89-3C',
 b'version': b'v1.0.0-test+0',
 b'power': b'1',
 b'': None}
```

7.3 Control via WebSocket

PIO devices implement a WebSocket server on TCP port 80 which is reachable at:

`ws://<PIO IP Address>:80/console`

PIO OS allows up to 8 simultaneous WebSocket connections. When all connection slots are in use new connections are rejected with error code “503 Service Unavailable” in the HTTP response.

Commands are transferred as plain text in UTF-8 format and must follow the syntax described in chapter 5, that means they consist of a device address (`101` for the PIO itself and `0` to `100` for a camera device) plus command string and parameters and are terminated by CRLF (“\r\n”):

`<ID> <command> <parameters><CRLF>`

For an example the command to get the network status of the PIO would be:

`“101 eth ipconfig\r\n”`

Or to set the video mode of a camera with ID `1` use:

`“1 video mode 12\r\n”`

PROTON devices always reply with an `OK` or `FAIL` message after a command has been processed. In the meantime, no additional command must be sent to the camera.

7.3.1 Multi-Client Operation

Since PIO OS allows multiple clients to access the attached camera(s) in parallel the incoming commands must be serialized so that only one client uses the RS485 interface at any point in time. This is done via a “first come, first serve” command queue as shown in the sequence diagram below.

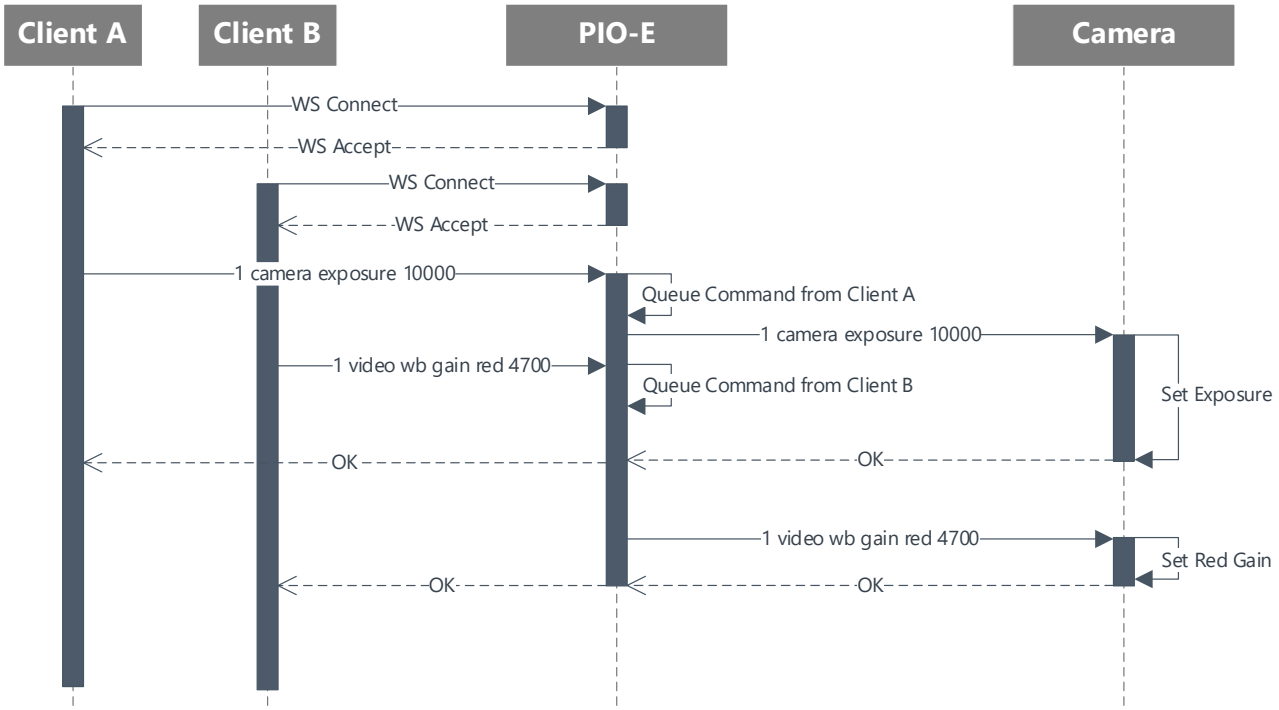

Figure 9: Sequence diagram showing the command queue concept

While the recommended workflow for clients is “send command, wait for reply” PIO OS does allow clients to queue multiple commands which will then be sequentially executed so the client will receive the replies in the same order the commands were sent. In this case the client is responsible for mapping the replies to the correct command.

Warning: When multiple clients control the same camera make sure that they do not override each other’s settings (e.g. both controllers try to set the exposure at the same time). It is recommended to split the workflow between the controllers, e.g. one controller is used for shading while the other is used for lens control.

7.3.2 Timeout Handling

When the connected camera is powered down or disconnected or a command is misspelled or gets corrupted during transmission, the PIO will never receive a reply from the camera. To avoid permanently locking up the command queue a configurable timeout is used. With the PIO command `system timeout` each client can configure how long the command handler shall wait for a reply before returning with a timeout error (error code -116).

It is the responsibility of the client to configure a sufficiently long command timeout before sending a command. The default timeout is 2 seconds, which is sufficient for most commands. Some operations like calibration tasks may take a lot longer though. Consult the “Command Reference” chapter in the camera’s reference manual for the advised timeout values for each command.

The example below shows usage of the timeout command to run the `bpc calibrate` command (bad pixel calibration) which can take up to 60 seconds to complete. Afterwards the client reverts the timeout to the default of 2 seconds and tries to change the video mode. In this example the camera gets disconnected while processing the command which causes the PIO-E to time out and send an error code to the client.

Note that the configurable timeout has no effect on PIO commands which are received at address `101`. As these are handled internally, they can never block due to an external device not responding.

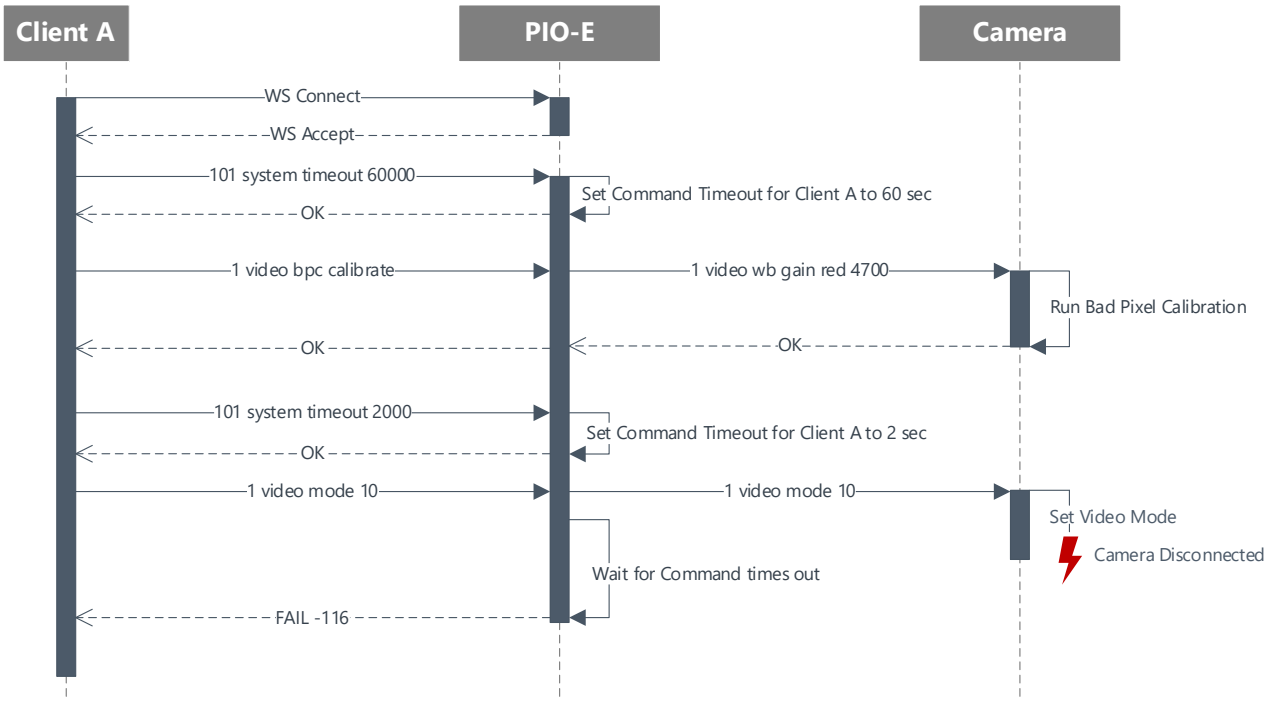

Figure 10: Sequence diagram showing command timeout usage

7.3.3 Error Codes

The following table shows the different errors that can occur when sending a command via a WebSocket connection:

Table 4: WebSocket command error codes

Error	Reply	Description
Queue Full	`FAIL -16`	The command queue is full (no more memory to store new commands). The command is dropped and must be resent later.
Invalid Syntax	`FAIL -96`	The command is not correctly terminated with a newline character (“\n”). The command is dropped.
Timeout	`FAIL -116`	The command was sent to the camera, but the camera did not reply within the configured timeout. Either the timeout was too short, or the camera is no longer available.
Execution Failed	`FAIL <error>`	The command was executed but during execution an error occurred: - If the command was a camera command this is the error code returned by the camera. Check the camera’s reference manual for details. - If it was a PIO command, see chapter 5.6 for a list of common error codes.

7.3.4 Finding and Controlling Camera Devices

Once a WebSocket connection to a PIO device has been established the following steps are recommended to find and control camera devices which are connected to the RS485 port:

- Flush the RS485 interface to make sure connected devices do not have any stale data in their receive buffers:

  → `101 system flush`
  ← `OK`

- Set the timeout to 1 second and set the `wait` flag which forces the command handler to wait for the specified timeout for new data to be received. Afterwards send the `identify` command to detect

connected cameras (see camera reference manual for details). In the following example there are two cameras connected with IDs `1` and `2`.

```
→ 101 system timeout 1000 1
← OK
→ 100 system identify
← id: vega 1 0 0 Camera A
← OK
← id: vega 2 0 0 Camera B
← OK
```

- Now set the timeout depending on the command you want to send without the `wait` flag being set. Using the default of 2 seconds is fine for most commands, e.g. to set the video mode of the camera with ID `2`:

```
→ 101 system timeout 2000
← OK
→ 2 video mode 5
← OK
```

7.3.5 Firmware Update

7.3.5.1 PIO Updates

Before starting a firmware update of the PIO device itself, it must be switched into update mode using the `system update` command. This is done automatically by the PROTON Updater or PROTON Control applications.

Once the PIO receives this command it gracefully disconnects all WebSocket clients (except the current client which requested the update) and rejects any new WebSocket connections. It also stops forwarding any data (IP and USB) to the RS485 camera interface.

If the client disconnects for some reason it may reconnect within 1 minute and resume the update. If the PIO does not receive update data for more than 1 minute, it will automatically reboot. This discards any stale update data and resumes normal operation so that the device becomes reachable again in case a client dies without properly closing the WebSocket connection.

If a new client tries to connect to a device in firmware update mode, the connection is rejected with error code “503 Service Unavailable” in the HTTP response. Alternatively, the client can check this without having to establish a WebSocket connection by evaluating the `update` property in the mDNS response (see chapter 7.2).

7.3.5.2 Camera Updates

When updating an attached camera device through the PIO the process is the same as when having the camera directly attached via a serial port adapter. For details, please check the reference manual of the camera device.

The only difference is that there can now be multiple clients accessing that camera. If an update is performed on a camera device that is also used by a different client, commands to that camera will fail until the update is finished. Newer cameras also report if they are in update state via the `system status` command which can be checked by the client.

7.3.6 Software Libraries

The following WebSocket implementations are recommended:

- C++: https://github.com/zaphoyd/websocketpp
- C#: https://learn.microsoft.com/en-us/dotnet/api/system.net.websockets.clientwebsocket
- Python: https://websockets.readthedocs.io

For an extensive list of WebSocket libraries and tools visit the Awesome WebSockets page.

7.3.7 Python Example

The following example shows how to open a WebSocket connection to a PIO device and fetch the current network state via the `eth ipconfig` command. It requires the “websockets” package which can be installed with pip:

```
pip install websockets
```

Note that the IP address is hard coded to `192.168.42.33`, it could be detected automatically by combining this example with the mDNS example in chapter 7.2.2.

Caution: This is a minimal example without any meaningful error handling.

```python
#!/usr/bin/env python

"""Connect to PIO device and check network status."""

from websockets.sync.client import connect
from websockets import exceptions

def check_network_status():
    with connect("ws://192.168.42.33:80/console") as websocket:
        # Send "eth ipconfig" command to PIO
        print("Sending command '101 eth ipconfig'")
        websocket.send("101 eth ipconfig\r\n")

        # Receive answer which may come in multiple messages
        reply = ""
        while(1):
            try:
                # Wait for max 1 second for the device to reply
                reply += websocket.recv(1)
                # Check if reply is complete
                if ("OK" in reply or "FAIL" in reply):
                    break
            except TimeoutError:
                print("Waiting for reply timed out")
                return
            except exceptions.ConnectionClosed:
                print("Connection got closed")
                return
            except:
                print("Unexpected error")
                return

        # Print reply
        print("Got reply:")
        print(reply)

if __name__ == "__main__":
    check_network_status()
```

7.4 Ping (ICMP Echo)

PIO devices respond to ping (ICMP echo) requests. This can be used to check if a device is available on the network without having to establish a WebSocket connection (this is e.g. used by PROTON Control to check if a PIO device is up again after a firmware update).

On Windows, Linux and macOS you can use the standard ping command by specifying the device's IP address or hostname (if there is a DNS server on the network):

```
ping <IP address / hostname>
```

Examples:

```
ping 192.168.1.22
ping PIO-E
```

See chapters 9.3.2 and 9.4.1 for details on how to set and get the hostname.

8 USB Control

The USB service port can be used for debugging and local control of the device. It is especially useful for initial configuration of the device, e.g. when there is no DHCP server in the network and you want to configure a static IP address.

Connecting the PIO-E via the USB-C port creates a virtual serial port on your PC or Mac which can be opened with any terminal program (see chapter 4.2) as well as PROTON Control (see chapter 4.1). The virtual serial port has the following properties:

- Baud rate is "don't care"
- 8-bit data, no parity, 1 stop bit (aka 8BitN1)
- No HW flow control

The command syntax is as described in chapter 5.

8.1 Timeout Handling

By default, commands sent via USB are treated the same as WebSocket commands, that means they are scheduled for execution as soon as the terminating CRLF ("\r\n") sequence is received. The PIO then waits until any ongoing command is finished before executing the command. While the USB command is executed, commands received via WebSocket are delayed so that USB and IP commands cannot get mangled.

The timeout handling is identical to the WebSocket implementation described in chapter 7.3.2 that means the controller is responsible to configure a sufficient command timeout with the `system timeout` command. If a command times out error code -116 is returned and the PIO continues with the next command from the command queue.

8.2 Firmware Update

8.2.1.1 PIO Updates

Before starting a firmware update of the PIO device itself, it must be switched into update mode using the `system update` command. This is done automatically by the PROTON Updater or PROTON Control applications.

Once the PIO receives this command it gracefully disconnects all WebSocket clients and rejects any new WebSocket connections. It also stops forwarding data received on the USB port to the RS485 camera interface.

If the update is interrupted for some reason, it may be resumed within 1 minute. If the PIO does not receive update data for more than 1 minute, it will automatically reboot. This discards any stale update data and resumes normal operation so that the device becomes reachable again via IP.

8.2.1.2 Camera Updates

When updating an attached camera device through the PIO the process is the same as when having the camera directly attached via a serial port adapter. For details, please check the reference manual of the camera device.

The only difference is that there can now be active WebSocket clients which are accessing the camera. If an update is performed on a camera device that is also used by a WebSocket client, commands to that camera will fail until the update is finished. Newer cameras also report if they are in update state via the `system status` command which can be checked by the client.

8.3 Debug Mode

Using the `system debug` command the USB port can be switched into a special debug mode. In debug mode the behavior of the USB interface changes:

- The port now mirrors all traffic of the RS485 interface which can be used to check which commands are sent via IP and how the camera replies.
- Data send via USB is now transmitted asynchronously, that means each character is forwarded to the RS485 interface immediately and there is no timeout handling. This allows the USB port to be used for interactive usage.

Debug mode is automatically disabled once the PIO is power cycled or rebooted.

Caution: Sending data via USB while the PIO is in debug mode should only be done if there is no IP client connected, otherwise the IP and USB commands can get mangled!

9 Command Reference

The following chapters document all commands in detail. Each sub-section describes a command group.

Notes regarding the command tables:

- If a command has no alias, it will be indicated by a forward slash (/).
- Commands that have a getter function reply with their command name first if they are called with no parameters or the required number of parameters for the getter function. Commands that do not have a getter function either reply with a special string (without sending the command name first) or do not produce any output at all despite the final `OK` / `FAIL` delimiter. For details on the command types see chapter 5.5.
- Most commands that change a setting have a default value. This is the value which the setting is reset to when calling the `settings reset` command.
- In the command syntax parameters are written in angle brackets, e.g. `<parameter name>`, and optional parameters are additionally wrapped in round brackets like `(<optional parameter>)`.
- Most of the commands can be executed almost immediately (in less than 100 ms). For those commands no runtime is specified. If the command needs longer to process this is stated in the third line of the command table.

9.1 General Commands

These commands are called without any parent command. They control basic shell functionality or print help messages.

9.1.1 alias

Command	`alias`		
Alias	/	**Type**	Special
Description	Print a list of all available command alias.		

9.1.2 help

Command	`help`

Alias	/	**Type**	Special
Description	Print the top-level help message which lists basic help instructions and a list of the top-level commands.		

9.1.3 history

Command	`history`		
Alias	/	**Type**	Special
Description	Print a list of the recently used commands.		

9.1.4 resize

Command	`resize`		
Alias	/	**Type**	Special
Description	Resize terminal output to current terminal window width.		

9.1.5 firmware

The firmware commands are used during the firmware update process and should normally not be executed manually by the user. The `firmware` command itself does not have any functionality, see subcommands below.

9.1.5.1 firmware list

Command	`firmware list`		
Alias	/	**Type**	Special
Description	List current firmware configuration.		

9.1.5.2 firmware request_upgrade

Command	`firmware request_upgrade <image_id>`		
Alias	/	**Type**	Direct
Description	Request upgrade of the image with the given ID. This command must be used *after* uploading a new image to the device, otherwise the update will not be performed.		
Parameter	`image_id`		
Type	Unsigned Integer		
Description	ID of the image that shall be upgraded.		
Valid Values	0: Upgrade software image (PIO OS)		

9.2 Settings Commands

These commands control the handling of device settings. They are called with the `settings` command prefix.

9.2.1 settings reset

Command	`settings reset <all>`		
Alias	`reset_settings`	**Type**	Direct
Description	Resets all settings to the default value. By default, only non-critical settings are reset. If all settings shall be reset (including the device name) call the command with `all` as shown below. Executing this command will reset the IP configuration so the device will shortly be disconnected from the network and then try to reconnect with its default settings.		

Examples:

```
→ 101 settings reset          # Reset non-critical settings.
← OK
→ 101 settings reset all      # Reset all settings.
← OK
```

Note: For a list of all critical settings that are reset by the `reset all` command, see chapter 6.1.

9.3 System Commands

These commands control basic system functionality like:

- System information
- Device name
- RS485 configuration
- Camera power
- …

They are called with the `system` command prefix.

9.3.1 system info

Command	`system info`		
Alias	`version`	**Type**	Special
Runtime	< 200 ms		
Description	Get system information.		

Example Output:

```
platform                : pioe
device name             : PIO-E
serial number           : CC-BA-97-B9-2D-84
software version        : v1.1.0
```

Notes on the output:

- `platform`: The platform string is unique for each PROTON device. It can be used by a controller to determine which device type it is talking to.
- `device name`: Can be set by the user with the `system name` command.
- `serial number`: Unique device serial number. The serial number is also the device's MAC address and is lasered to the bottom of the device.
- `software version`: PIO OS version.

Table 5: Supported Devices

Platform String	**Product**
pioe	PROTON PIO-E

9.3.2 system name

Command	`system name <name string>`		
Alias	`name`	**Type**	Getter, Setter
Description	Set device name. The name string may contain up to 5 words which in total (and including white spaces) have a length of 32 characters. The device name is also used as the DNS hostname. Since hostnames cannot contain space characters, they are replaced by underscores (e.g. if the device name is 'my name' the hostname will be 'my_name').		
Parameter	`name string`		
Type	String		
Description	Device name to set, max 32 characters.		

9.3.3 system runtime

Command	`system runtime`		
Alias	`runtime`	**Type**	Pure Getter
Description	Print device runtime since boot in seconds. The counter gets reset by a power cycle or reboot.		

9.3.4 system reboot

Command	`system reboot`		
Alias	`reboot`	**Type**	Direct
Runtime	< 200 ms + time to reboot		
Description	Reboot the device.		

9.3.5 system update

Command	`system update`		
Alias	`update`	**Type**	Direct
Description	Put the device into firmware update mode which allows it to process firmware update messages. When entering update mode via USB all WebSocket connections are disconnected and new connections are rejected. When entering update mode via a WebSocket client, all other clients except the current one will be disconnected and new connections are rejected. This ensures that the update process cannot be interrupted by another client. Entering update mode also automatically disables debug mode if it is enabled (see `system debug` command). When in update mode all camera commands are rejected with error code -140. Firmware update mode can only be left via a reboot. If no new update data is received for 1 minute, the device automatically aborts the update and reboots to avoid getting stuck in update mode. After the reboot it is reachable as usual again via WebSocket. If the system is already in update mode, this command will return error code -120. To check if the system is currently in update mode use the `system status` command.		

9.3.6 system status

Command	`system status`		
Alias	`status`	**Type**	Special
Description	Get a short system status which can be either of: *down:* Network interface is down (no network connection). *static:* Device is up and uses a static IP address. *dhcp:* Device is up and has received an IP address via DHCP. *connected:* Device is connected to at least one WebSocket client (use `eth ipconfig` command to get number of active clients). *update:* Device is in firmware update mode (see `system update` command). *error:* Device is in error mode, use the `system error` command to get details.		

9.3.7 system error

Command	`system error`		
Alias	`error`	**Type**	Special
Description	Print error log. In case no errors are logged it only returns `OK`. The status LED blinks red when an error was logged and the `system status` command will also report that the system is in “error” state.		

9.3.8 system volatile

Command	`system volatile <value>`		
Alias	`volatile`	**Type**	Getter, Setter
Description	Set a 32-bit runtime variable which will keep its value until a reboot is performed. Can be used by a controller to store arbitrary information or check if device got rebooted (volatile value got reset to 0).		
Parameter	`value`		
Description	Volatile value to set.		
Min	0		
Max	4294967295 = 0xFFFFFFFF		
Default	0		

9.3.9 system ping

Command	`system ping`

Alias	`ping`	**Type**	Special
Description	Check if device exists and replies. This command does nothing except replying with “OK”.		

9.3.10 system baudrate

Command	`system baudrate <rate>`		
Alias	`baudrate`	**Type**	Setter, Getter
Description	Set RS485 baud rate. This baud rate is used to communicate with devices attached to the RS485 port and it must match the baud rate of the connected devices. To get supported rates use the `baudrate list` subcommand.		
Parameter	`rate`		
Type	Unsigned Integer		
Description	New RS485 baud rate to set.		
Valid Values	9600, 1440, 19200, 57600, 115200, 230400, 250000		
Default	115200		

9.3.10.1 system baudrate list

Command	`system baudrate list`		
Alias	/	**Type**	List
Description	List all supported baud rates of the RS485 interface.		

Note: Since this command does not print IDs, but explicit values, it does not use the leading hash (#) like other `list` commands.

Example:

```
→ 101 system baudrate list
← 9600
← ...
← 115200
← OK
```

9.3.11 system timeout

Command	`system timeout <ms> (<wait>)`		
Alias	`Timeout`	**Type**	Setter, Getter
Description	Set timeout for command processing in ms. When the PIO receives a camera command via IP it waits until either the device replies with `OK` or `FAIL` or the timeout has elapsed in which case error code -116 is returned. The minimum timeout is 0 ms (return immediately with timeout error, can be used to flush the device), the maximum is 120000 ms. The default timeout is 2000 ms which is a safe value for most commands. It is the responsibility of the controller to change the timeout before executing a long running command (e.g. defect pixel calibration), otherwise command execution will fail with error code -116 and following commands may also fail because the camera is still busy or replies while sending the next command which causes garbage on the RS485 interface. For short running commands the timeout may be reduced to increase controller responsiveness in case of errors. For recommended timeout values, consult the camera's command description in its reference manual. The timeout can be set for each client independently (the USB port is also considered as a client in this case). The PIO will always use the timeout which was last set by the client. When a new WebSocket client connects, the timeout is initially set to the default value. When the optional `wait` flag is set the PIO will always wait for the timeout to elapse before continuing with the next command. In this case the timeout error code -116 is not returned. This is needed for commands like the camera’s `identify` command which may return multiple replies (or no reply at all). If the `wait` flag is omitted, it is implicitly set to `false`. Note: If the USB port is in debug mode, the configured timeout is ignored, see `system debug` command for details.		

Parameter	`ms`	`wait`
Type	Unsigned Integer	Boolean
Description	New timeout value to set for this connection in ms.	Optionally: Wait until timeout has fully elapsed, regardless of the device's response.
Min	0 (return immediately with error -116)	0
Max	120000	1
Default	2000	0 (also used if omitted)

For details on how to use the timeout see the examples in chapter 7.3.

9.3.12 system flush

Command	`system flush`		
Alias	`update`	**Type**	Direct
Runtime	`< 200 ms`		
Description	Flush the command buffer of attached devices by sending two newline characters on the RS485 interface and discarding all data which is received from the attached devices afterwards. Should be used before searching for attached devices.		

9.3.13 system power

Command	`system power <enable>`		
Alias	`power`	**Type**	Setter, Getter
Description	Enable or disable power to external devices. Can be used to perform a power-cycle of the connected devices (you need to turn off and on in two commands). Note that powered down devices will not respond to any commands! **Caution:** This setting is stored in the persistent setting storage, so power will not be enabled by a power-cycle or reboot of the PIO. This allows use cases where the attached device shall stay disabled until it is needed to conserve power.		
Parameter	`enable`		
Type	Boolean		
Description	Turn power for attached devices on or off.		
Default	On		

9.3.14 system debug

Command	`system debug <enable>`		
Alias	`debug`	**Type**	Setter, Getter
Description	Enable or disable USB debug mode. In debug mode the USB port will mirror all traffic which is sent to attached devices via the RS485 interface including commands which are received via IP. Additionally, the USB port becomes fully interactive in debug mode, that means all characters which are received are immediately duplicated to the RS485 interface and there is no timeout handling. Enabling debug mode is only recommended for advanced users as sending commands while debug mode is enabled can interfere with the commands received via IP. This setting is not persistent that means after a reboot or power cycle debug mode is always disabled. Debug mode cannot be entered if the device is in update mode and entering update mode automatically disables debug mode (see `system update` command).		
Parameter	`enable`		
Type	Boolean		
Description	Enable or disable USB debug mode.		
Default	On		

9.4 Ethernet Commands

This command group controls the IP interface of the device. Its commands are called with the `eth` command prefix.

Caution: Using the wrong IP config can make the device unreachable. To recover from such a state either change the configuration via the USB service port (see chapter 2.3) or perform a device reset via the push button as described in chapter 2.5.2.

9.4.1 eth ipconfig

Command	`eth ipconfig`		
Alias	`ipconfig`	**Type**	Special
Description	Get current ethernet configuration.		

This command behaves like the “ipconfig” or “ifconfig” commands on Windows and Linux and dumps the current network configuration. Example Outputs:

Static IP:

```
status  : up
type    : static
mac     : CC-BA-97-B9-2D-84
address : 10.0.0.101/24
gateway : 10.0.0.1
hostname: PIO-E
clients : 1 / 8
```

DHCP:

```
status  : up
type    : dhcp
mac     : CC-BA-97-B9-89-3C
address : 192.168.42.33/24
gateway : 192.168.42.1
hostname: PIO-E
clients : 1 / 8
```

Notes on the output:

- `status`: Current link status, can be either `up` or `down`.
- `type`: Type of the active IP address (see below), can be either `static` or `dhcp`.
- `mac`: The device’s unique MAC address.
- `address`: Currently active IP address of the device (the address on which it is reachable).
- `gateway`: Currently active standard gateway of the device.
- `hostname`: Hostname of the device. If there is a DNS server available in the network, the device is also reachable via this name in addition to its address.
- `clients`: The number of active WebSocket clients followed by the maximum number of connections (in the above example “clients: 1 / 8” means 1 active client, 8 connections max).

9.4.2 eth ip

Command	`eth ip <addr> <mask>`		
Alias	`ip`	**Type**	Setter, Getter
Description	Set static IP address and netmask. The `addr` parameter is the IPv4 address (e.g. 192.168.1.2), the `mask` is the shortened netmask, e.g. 24 for 255.255.255.0.		
Parameter	`addr`	`mask`	
Type	String	Unsigned Integer	
Description	IPv4 address	Shortened netmask	
Min	0.0.0.0	1 (= 128.0.0.0)	
Max	255.255.255.255	31 (= 255.255.255.254)	
Default	10.0.0.101	24 (= 255.255.255.0)	

9.4.3 eth gateway

Command	`eth gateway <addr>`		
Alias	`gw`	**Type**	Setter, Getter
Description	Set the standard gateway which is used when no gateway is set via DHCP. If there is no standard gateway in your network use '0.0.0.0'.		
Parameter	`addr`		
Type	String		
Description	IPv4 address		
Min	0.0.0.0		

Max	255.255.255.255
Default	10.0.0.1

9.4.4 eth dhcp

Command	`eth dhcp <enable>`		
Alias	`dhcp`	**Type**	Setter, Getter
Description	Enable or disable DHCP client. If enabled the static IP address will be overridden with a dynamic IP address as soon as one is received from a DHCP server. When disabled only the static IP is used.		
Parameter	`enable`		
Type	Boolean		
Description	Enable or disable DHCP client.		
Default	On		

9.4.5 eth config

Command	`Eth config <addr> <mask> <gateway> <dhcp>`			
Alias	`config`	**Type**	Setter, Getter	
Description	Get or set all ethernet related settings with a single command.			
Parameter	`addr`	`mask`	`gateway`	`dhcp`
Type	String	Unsigned Integer	String	Boolean
Description	IPv4 address of device	Shortened netmask	IPv4 address of standard gateway	DHCP client enable
Min	0.0.0.0	1	0.0.0.0	Off
Max	255.255.255.255	31	255.255.255.255	On
Default	10.0.0.101	24	10.0.0.1	On

10 Alias Reference

This chapter lists all the available command alias.

The list below can also be generated by the device with the `alias` command.

Table 6: Command alias overview.

Alias	Full Command Name
ipconfig	eth ipconfig
ip	eth ip
gw	eth gateway
dhcp	eth dhcp
config	eth config
reset_settings	settings reset
info	system info
name	system name
runtime	system runtime
reboot	system reboot
update	system update
status	system status
error	system error
volatile	system volatile
ping	system ping
baudrate	system baudrate
timeout	system timeout
flush	system flush

Alias	Full Command Name
power	system power
debug	system debug

Appendix A: Software Licenses

PIO OS makes use of software licensed under the following conditions.

Apache License Version 2.0

Parts of PIO OS are distributed under the Apache License Version 2.0:

means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and

wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Appendix B: Document Revision History

Revision	Date	Chapter	Changes
v1.0.0	03. Sep. 2025	All	Initial release.
v1.0.1	10. Sep. 2025	4.1	Added more details and examples on how to use PROTON Control with a PIO device.
		9.4.3	Added note to use “0.0.0.0” for “no gateway”.
v1.1.0	03. Mar. 2026	All	Updated reference manual for multi-client support.
		2.1, 2.2.2	Added maximum power draw and output. Removed reference to passive PoE injectors to avoid confusion since those usually use 12 or 24 V and PIO-E only supports 48 V.
		8	Added dedicated chapter on USB control.
		9.3.1	Updated platform name from “pio2” to “pioe”.
		9.3.5, 9.3.6, 9.3.7, 9.3.14, 9.4.1	Added `system status` and `system debug` commands, updated `system update`, `system error` and `eth ipconfig` commands.
		9.3.10	Updated alias of `system baudrate` command from `rs485_baud` to `baudrate`.
		9.3.11, 9.3.12	Added new `system timeout` and `system flush` commands.
		10	Updated alias list.