



Programmable Attenuator User Manual



Table of Contents

1. Introduction.....	4
2. Safety Instructions	5
3. Product Overview.....	5
4. Setting Up Your Device	7
5. Local Mode	7
6. Remote Mode	8
7. Troubleshooting	9
8. Maintenance & Cleaning	9
9. Specifications	10
10. Warranty & Support Information	11
11. MATLAB Function Script.....	11



1. Introduction

Before using the QPE, please read this manual thoroughly to ensure safe and efficient usage.

The QuinStar QPE-XXXXXX model is a motorized programmable rotary vane attenuator for use in millimeter wave systems. It is available in most standard frequency bands and has the following key features:

- **User selectable calibration maps:**

- o Each unit comes with several frequency maps, each detailing specific motor positions tied to an attenuation value. This configuration provides a steady and predictable attenuation values across all frequencies.

For instance, the KA-band version of the QPE has an insertion loss of 0.3dB at 33GHz. Thus, the 0dB attenuation state at 33GHz, as well as the 0dB state for any other frequencies outlined in the unit's calibration table, will all reflect a consistent 0.3dB insertion loss. Similarly, the 1dB attenuation state across all specified frequencies will indicate a total of 1.3dB insertion loss, and so forth.

Please be aware that changing calibration maps necessitates a remote connection for configuration. By default, the selected calibration map is situated at the midpoint of the operational frequency band. Please review your unit's calibration for more information.

- **High accuracy, up to +/-0.01dB attenuation resolution:**

- o Consider, once more, the KA-band version of the QPE that has a base insertion loss of 0.3dB. The achievable attenuation states and their corresponding insertion losses are as follows:

0.00dB attenuation results in a 0.30dB insertion loss.

0.01dB attenuation results in a 0.31dB insertion loss.

0.02dB attenuation results in a 0.32dB insertion loss.

... and so on.

Be aware that as attenuation increases, accuracy may vary. It's crucial to thoroughly examine your specific calibration tables to maintain accuracy expectations.

- **High repeatability, up to +/-0.01dB error:**

- o Each unit ensures high repeatability to guarantees the integrity of the pre-mapped calibration tables with each power cycle. Note that as attenuation increases, not only does accuracy fluctuate, but repeatability error may also rise. It's essential to closely review your designated calibration tables to set proper error expectations.

- **Manual and remote operation modes:**

- o All QPE units are equipped with a numeric keypad, allowing users to manually adjust attenuation states at the device. Additionally, each unit is configured with multiple I/O ports for remote functionality via RS485, USB-C, and LAN connections. For remote operations, the QPE communicates using SCPI format commands and responses. Refer to the remote section for detailed operational instructions.

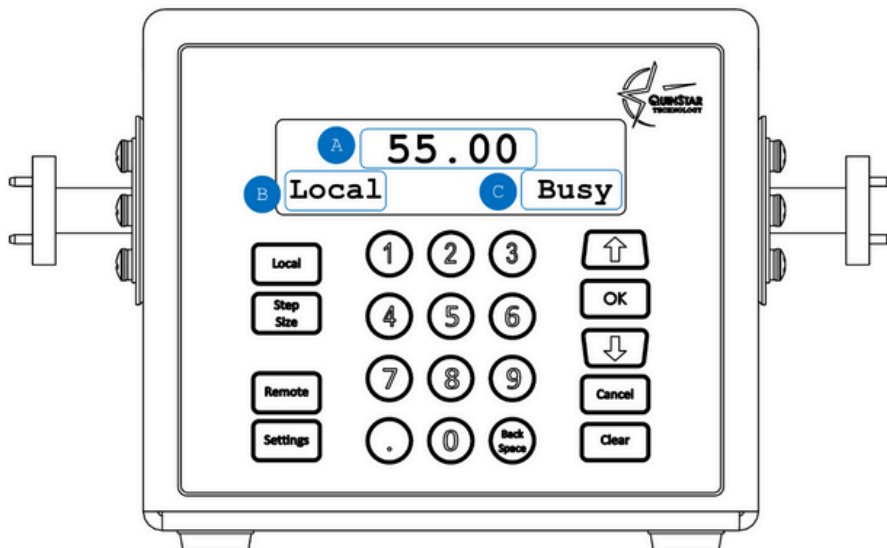
2. Safety Instructions

- **Standard Lab Use Only:** The QPE is designed exclusively for use in a standard laboratory setting, with humidity levels between 30-70% and temperatures ranging from 20°C to 25°C (68°F to 77.0°F)
- **RF Power Warning:** Please ensure that the RF input power does not surpass the limits outlined in the specification table. Exceeding these limits can damage the internal components responsible for attenuation, potentially necessitating a part replacement and a recalibration. If such an issue arises, reach out to QuinStar promptly for servicing.
- **Power Supply Warning:** This unit is supplied with a 24V, 2.1A adapter. Utilizing an adapter outside of these specifications can damage the device.
- **Low Attenuation Warning:** QPE demonstration units return to the lowest attenuation state (0 dB) after every power cycle. This is a necessary step for internal motor calibration. Exercise caution when using a QPE demonstration unit in a system with strict power handling requirements.

3. Product Overview

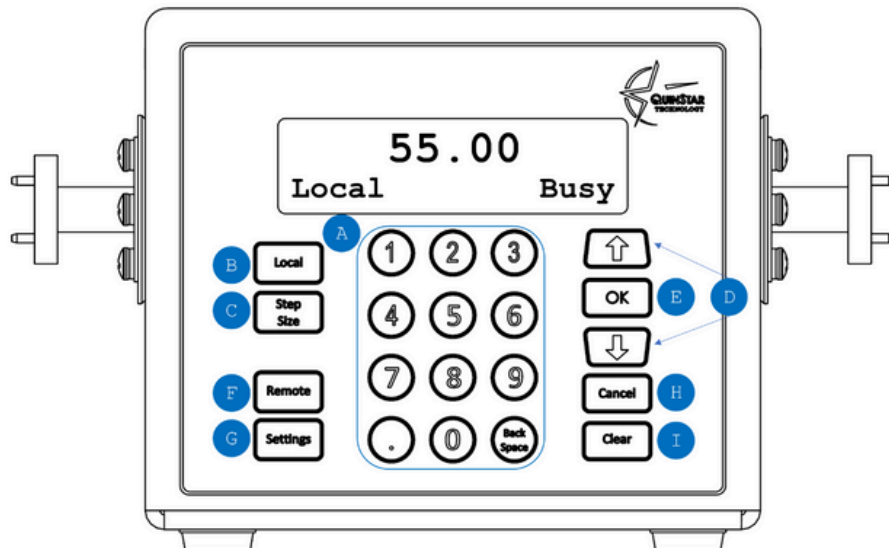
3.1 Front Panel:

1. Display:



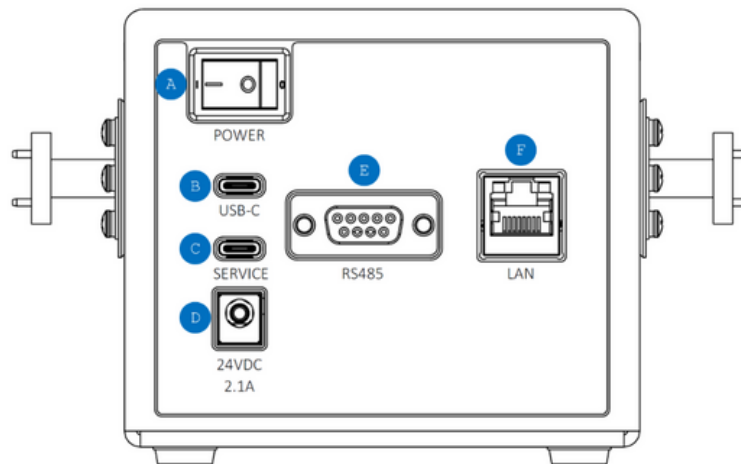
- a) Attenuation setting.
- b) Chosen mode: Either Local or Remote. Remote modes encompass RS485, USB, and LAN connections.
- c) Indicates motor activity or baud rate:
 - a. “Busy” implies the motor is adjusting to the set attenuation level.
 - b. “9600” or other denotes the selected baud rate for the RS485 connection.

2. Buttons:



- a) Number Keypad: Facilitates manual entry of attenuation value in Local mode.
- b) Local Mode: Engages the Local mode and activates the number keypad for input.
- c) Step Size: Adjusts the incremental value for the Up and Down functions.
- d) Up/Down Buttons:
 - In Local mode, increases or decreases the current attenuation value by the defined step size.
 - In RS485 mode, modifies the baud rate setting for the RS485 interface.
- e) OK:
 - In Local mode, confirms the inputted attenuation value.
 - In RS485 mode, during the baud rate selection process, the OK button confirms the displayed baud rate. Refer to the 'Setting Baud Rate' section for further details.
- f) Remote: Toggles between different remote connection options: USB, RS485, and LAN.
- g) Settings: When in RS485 mode, this button launches the baud rate selection wizard. For details, see section XX.
- h) Cancel: Cancels any input selection wizard.
- i) Clear: Clears value in any input selection wizard.

3.2 Back Panel & Remote Communication Type:



- a) Power Button: main power switch.
- b) USB-C: Functions as a USB 2.0 CDC device port
- c) Service port: Do not use. This port is used for QuinStar servicing only.
- d) DC Barrel: 24V, 2.1A.
- e) LAN: Telnet.
- f) DB9: TIA/EIA RS485. See pinout details.

4. Setting Up Your Device

1. Ensure the power button is in the OFF position.
2. Connect the 24V/2.1A power supply to the DC barrel located at the rear of the unit.
3. Flip the power button to the ON position.
4. Wait until the QPE's busy status indicator vanishes. During startup, the unit executes a homing sequence to establish the primary reference point essential for calibration table use.
5. Once the QPE's busy status disappears and screen shows a "Local" mode prompt, the unit is fully prepared for operation.

5. Local Mode

In this level, there are two methods to manually input your desired attenuation value:

1. **Precise Value Entry:** Initiate the attenuation value wizard by pressing any numbers from 0 to 9, as well as the '.' (decimal point). Once the exact value is entered, press the OK button to confirm the selection.
2. **Incremental Value Adjustment:** By pressing the UP and DOWN keys, the attenuation value wizard will activate while incrementing or decrementing the value by the unit's preset step size. To customize the step size, simply press the STEP SIZE button at any time.

Upon a cold boot, the device defaults to an attenuation value of 50dB and utilizes the default calibration lookup table uploaded to the unit (typically close to the center frequency of the band). To change the calibration lookup tables, the remote command `FREQ XX.XX` must be issued, where `XX.XX` represents the frequency of the desired calibration lookup table.



6. Remote Mode

6.1 Connector Types & Protocols:

On boot, the device initiates a homing sequence to establish a primary reference point that the calibration tables rely on. To enable remote connections, please select the appropriate remote mode. This can be accomplished by cycling through the modes using the REMOTE MODE button. Trying to transmit data packets through an unselected remote session will lead to a failed connection and will fill the buffer for that connection. This can result in erroneous results, especially during the first command sent through that port. **Connection Options:**

- **LAN:** Enables Telnet communication over the local area network (LAN). All QPE units are equipped with the Lantronix XPort Edge XPE200100S LAN module, which has a static IP address of 192.168.1.123/24 and a default administrator account “admin” with password “QPE2023!”. For any customizations, please consult the Lantronix user guide [here](#).
- **DB9:** Provides a TIA/EIA RS485 connection. Toggle the Settings button to cycle through the desired baud rate. Refer to the Specification section for pinout details.
- **USB-C:** Functions as a USB 2.0 CDC device port. No drivers are necessary for Windows 10 or Windows 11, Windows will recognize device as serial USB device with name “Enhanced QPA”

Please note that while the connectors and protocols may vary, commands listed in the Data Command Definitions section are universal to all connector options.

6.2 Data Command Definitions:

Note all commands listed must be terminated with a Carriage Return (CR) followed by a Line Feed (LF). This is equivalent to ASCII \r (0x0D) and ASCII \n (0x0A), respectively.

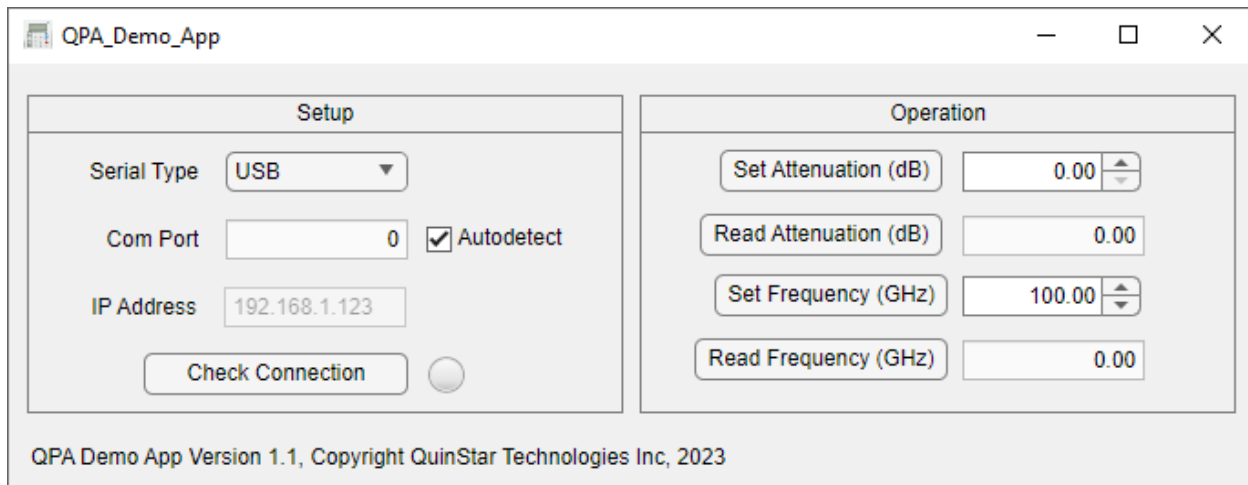
The commands below are separated into 3 families: attenuation, system, and manufacturing. The attenuation commands are the typical commands most will use to control the attenuation properties of the device. The system commands provide utilities for programmatic control, validation, and error handling. The manufacturing commands provide unit-specific information necessary for servicing the equipment.

Attenuation commands	Example	Comments
ATTEN XX.XX	ATTEN 80.00	Sets attenuation value to 80.00dB.
ATTEN?	ATTEN?	Reads the current attenuation value.
FREQ XX.XX	FREQ 36.50	Sets the reference calibration table to 36.50GHz.
FREQ?	FREQ?	Reads frequency of selected calibration table.

System commands	Comments
SYST:ERR?	Returns error(s) if any, starting from the oldest in queue.
SYST:FAULT:LOG?	Currently not in use.
*OPC? *STB?	Returns 1 if unit has executed all pending SCPI commands.
*IDN? *RST	Currently not in use.
	Returns the QPE description along with its build version.
	Software reset, equivalent to a physical power cycle.

Manufacturing commands	Comments
MFG:SERNUM?	Returns the serial number of the unit.
MFG:FREQ:MIN?	Returns the lowest frequency of the available calibration tables.
MFG:FREQ:STEP?	Returns the frequency step size of the next available calibration table.
MFG:FREQ:DEFAULT?	Returns the frequency of the default calibration table used at startup.
MFG:ATTEN:DEFAULT?	Returns the attenuation value used at startup.
MFG:UID?	Returns the UID of the unit, required during servicing.

6.3 QPE Demo Application



A straightforward demonstration application can be found on the QuinStar website. This application provides the means to set and read attenuation and frequency values. It utilizes the same MATLAB function script located at the end of this user manual. Be sure to verify the connection settings (as shown in the “Setup” section) before transmitting any set or read commands to the device.

7. Troubleshooting

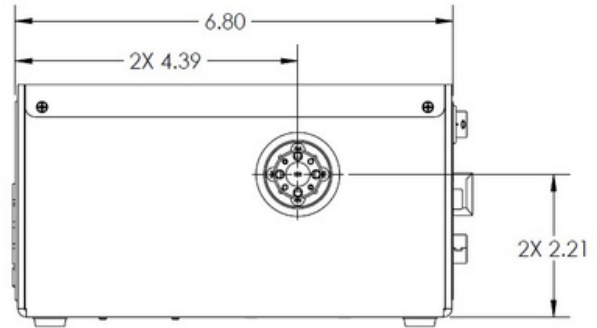
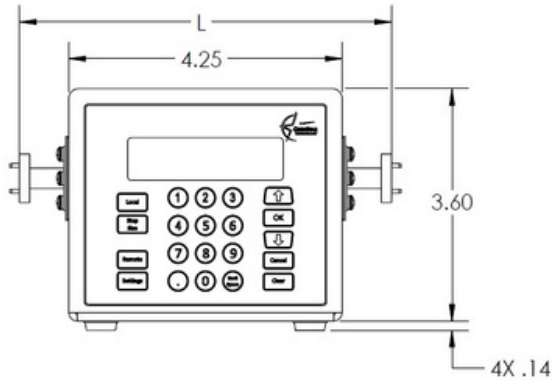
- Issue: Device won't turn on
 - Solution: To address this issue, inspect the power connections and verify the outlet's proper functioning. Additionally, ensure that the power supply is providing the correct output at the DC barrel jack, which should be 24V and capable of delivering up to 2.1A.
- Issue: Noreponse from remote connections
 - Solution: For this problem, verify compatibility and secure connection of the cable and connector. Make sure the correct remote mode is selected by cycling through the options using the REMOTE MODE button.

8. Maintenance & Cleaning

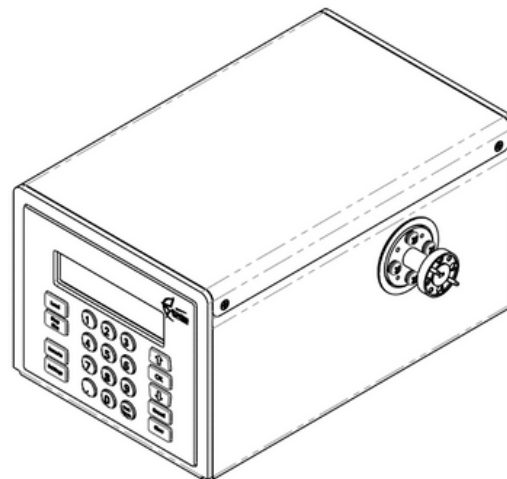
- Ensure that RF waveguide input and output are free from dirt or debris.
- Prior to cleaning, power off the device and disconnect it.
- Utilize a soft, damp cloth for cleaning. Avoid abrasive materials to prevent any damage to the input or output face of the RF waveguides.

9. Specifications

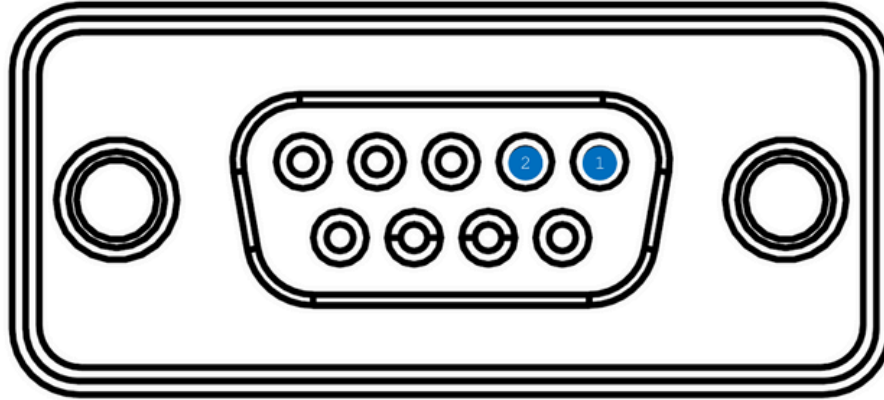
9.1 Dimensions



BAND	L	FLANGE
K	8.48	UG-595/U
Ka	6.92	UG-599/U
Q	6.58	UG-383/U (ANTI-COCKING)
U	6.38	UG-383/U-M (ANTI-COCKING)
V	5.78	UG-385/U (ANTI-COCKING)
E	5.78	UG-387/U (ANTI-COCKING)
W	5.78	UG-387/U-M (ANTI-COCKING)
F	5.78	UG-387/U-M (ANTI-COCKING)
D	5.78	UG-387/U-M (ANTI-COCKING)



9.2 Serial RS485 pinout



1. RS485 B (-)
2. RS485 A (+)

10. Warranty & Support Information

This device comes with an X days of warranty.

For technical support, please contact:

QuinStar Technology Inc
(310) 320-1111
support@quinstar.com
<https://www.quinstar.com>

11. MATLAB Function Script

The user functions outlined in the MATLAB function script require a QPE object, which specifies the communication port to be used for interfacing with the physical QPE device.

Initialization function	Example	Purpose
<pre>QPE(comPortOrLanAddr, serialType, terminator)</pre>	<pre>QPE = QPE(comPort, serialType);</pre>	<p>Initializes communication with the QPE device. This object is passed to the user functions function below.</p> <p>comPortOrLanAddr: A string containing the COM port value for the RS485 port or the LAN's IP address (e.g., "1", "192.168.1.123").</p> <p>serialType: A string describing the connection type (valid values: "USB", "RS485", "LAN").</p> <p>terminator: A string (optional) that defines the termination type if different from the default values.</p>



User function	Example	Notes
<code>set_att(obj, att)</code>	<code>set_att(QPE, 50.01);</code>	Sets attenuation to 50.01dB.
<code>read_att(obj)</code>	<code>read_atten(QPE);</code>	Returns the current attenuation value.
<code>set_freq(obj, freq)</code>	<code>set_freq(QPE, 60);</code>	"Selects the 60GHz calibration table. The default is set to the center of the working frequency."
<p>Note: After setting the frequency, a subsequent "set_att" command must be sent to adjust the motor position corresponding to the attenuation level at the new frequency. Simply setting the frequency will not automatically readjust the current attenuation level."</p>		
<code>read_freq(obj)</code>	<code>read_freq(QPE);</code>	Returns the selected frequency cal table.
<code>read_freq_bin(obj)</code>	<code>read_freq_bin(QPE);</code>	Returns the current frequency bin selected in memory; used for troubleshooting purposes with QuinStar Application Engineer.
<code>read_mfg_sn(obj)</code>	<code>read_mfg_sn(QPE);</code>	Returns the serial number of the unit.
<code>read_mfg_min_freq(obj)</code>	<code>read_mfg_min_freq(QPE);</code>	Returns the minimum frequency of the unit.
<code>read_mfg_freq_step(obj)</code>	<code>read_mfg_freq_step(QPE);</code>	Returns the frequency step between each freq cal table.
<code>read_mfg_default_att(obj)</code>	<code>read_mfg_default_att(QPE);</code>	Returns the default attenuation at boot, not user configurable.
<code>read_system_error_query(obj)</code>	<code>read_system_error_query(QPE);</code>	Returns system errors, if any.
<code>read_system_fault_log_query(obj)</code>	<code>read_system_fault_log_query(QPE);</code>	Returns fault logs, if any.
<code>query_operation_complete_status(obj)</code>	<code>query_operation_complete_status(QPE);</code>	Returns '1' if all QPE operations are complete and ready for another command, '0' if otherwise.
<code>query_status_byte(obj)</code>	<code>query_status_byte(QPE);</code>	Returns the status byte.



```
classdef QPE < handle

    properties
        serialType
        comPort
        baudRate
        dataBits
        parity
        stopBits
        flowControl
        byteOrder
        lanAddr
        lanPort= 23
        terminator = "CR/LF"
        timeout= 10
    end

    methods
        function obj = QPE(comPortOrLanAddr, serialType, terminator)
            % Initialize with COM = 0 to autodetect
            if ~exist("terminator", "var")
                obj.terminator = "CR/LF";
            else
                obj.terminator = terminator;
            end
            % Set appropriate serial settings
            obj.serialType = serialType;
            if obj.serialType == "USB"
                obj.baudRate = 19200;
                obj.dataBits = 8;
                obj.parity = 'none';
                obj.stopBits = 1;
                obj.flowControl = 'none';
            elseif obj.serialType == "RS485"
                obj.baudRate = 9600;
                obj.dataBits = 8;
                obj.parity = 'none';
                obj.stopBits = 1;
                obj.flowControl = 'none';
                obj.byteOrder = "big-endian";
            elseif obj.serialType == "LAN"
                obj.lanAddr = comPortOrLanAddr;
                obj.write("*IDN?");
                obj.test_conn()
            else
                error("Invalid IP Address specified.");
            end
        end
    end
end
```

```

        end
    end
    % Set COMPort
    if obj.serialType == "USB" || obj.serialType == "RS485"
        if comPortOrLanAddr == 0
            obj . detect _COM ();
        else
            obj.comPort= strcat("COM", int2str(comPortOrLanAddr));
        end
    end
end

%%% COMMUNICATION COMMANDS

function set_baudrate(obj, baudrate)
    obj.baudrate = baudrate;
end

function set_ip_addr(obj, ip)
    obj.lanAddr = ip;
end

function detect_COM(obj)
    %% Automatically finds a valid QPECOMPort
    for port=0:256
        obj.comPort= strcat("COM", int2str(port));
        validPort=false;
        try
            if obj.test_conn()
                validPort = true;
                break;
            end
        catch
            continue;
        end
    end
    if ~validPort
        error("Valid port not found for QPE. Check connection.");
    end
end

function response=read(obj)
    if obj.serialType == "LAN" % Using TCP and Telnet
        % Carriage return and line feed characters for termination
        cr =string(char(hex2dec("0D")));
    end
end

```



```
lf =string(char(hex2dec("0A")));
% Setting up TCP Client
t =tcpclient(obj.lanAddr, obj.lanPort, "ConnectTimeout", 10);
configureTerminator(t,"CR","LF");
% Waiting for socket to populate with data before reading
start_time = datetime("now");
whileseconds(datetime("now")-start_time) < 10 && t.NumBytesAvailable == 0
    continue;
end
response = read(t, t.NumBytesAvailable, "string");
% Removing termination characters
if ~isempty(response)
    response = erase(response, cr+lf);
end
clear t;
else% Using direct serial comms for RS485 and USB
s =serialport(obj.comPort, obj.baudRate, 'DataBits', obj.dataBits, 'Parity', ...
    obj.parity, 'StopBits', obj.stopBits, 'FlowControl', obj.flowControl);
if obj.serialType == "RS485"
    s.ByteOrder = "big-endian";
end
configureTerminator(s, obj.terminator);
response = read(s, s.NumBytesAvailable, "string");
clear s;
end
end

functionwrite(obj, msg)
    if obj.serialType == "LAN" % Using TCP and Telnet
        cr =string(char(hex2dec("0D")));
        lf =string(char(hex2dec("0A")));
        t =tcpclient(obj.lanAddr, obj.lanPort, "ConnectTimeout", 10);
        configureTerminator(t,"CR","LF");
        write(t, msg+cr+lf, "string");
    else% Using direct serial comms for RS485 and USB
        s =serialport(obj.comPort, obj.baudRate, 'DataBits', obj.dataBits, 'Parity', ...
            obj.parity, 'StopBits', obj.stopBits, 'FlowControl', obj.flowControl);
        if obj.serialType == "RS485"
            s.ByteOrder = "big-endian";
        end
        configureTerminator(s, obj.terminator);
        writeline(s, msg);
        clear s;
    end
end
end
```



```
function response=query(obj, msg)
    if obj.serialType == "LAN" % Using TCP and Telnet
        % Writingstring cr
        =string(char(hex2dec("0D"))); lf
        =string(char(hex2dec("0A")));
        t =tcpclient(obj.lanAddr, obj.lanPort,"ConnectTimeout", 10);
        configureTerminator(t,"CR","LF");
        write(t, msg+cr+lf,"string");
        % Readingstringwith same tcpclientobject
        start_time=datetime("now");
        whileseconds(datetime("now")-start_time) <10 && t.NumBytesAvailable == 0
            con t inue;
        end
        response =read(t, t.NumBytesAvailable,"string");
        if ~isempty(response)
            response = erase(response, cr+lf);
        end
        clear t;
    else % Using direct serial comms for RS485 and USB
        s =serialport(obj.comPort,obj.baudRate,'DataBits', obj.dataBits, 'Parity', ...
            obj.parity,'StopBits',obj.stopBits, 'FlowControl', obj.flowControl);
        if obj.serialType == "RS485"
            s.ByteOrder = "big-endian";
        end
        configureTerminator(s, obj.terminator);
        response =writeread(s,msg);
        clear s;
    end
end

function result =test_conn(obj)
    response= obj.query("*IDN?");
    if isempty(response)
        result = false;
    else
        result = contains(response,"QuinStarTechnologies Inc");
    end
end

%%% USER COMMANDS

function set_att(obj, att)
    obj.write("ATTEN " + sprintf('%.2f', att));
    obj.query_operation_complete_status();
```



```
end

function response = read_att(obj)
    obj.query_operation_complete_status();
    response = str2num(obj.query("ATTEN?"));
end

function set_freq(obj, freq)
    obj.write("FREQ " + sprintf('%.2f', freq));
    obj.query_operation_complete_status();
    if obj.serialType == "LAN"
        obj.read(); % DEBUG LINE BUG FIX
    end
end

function response = read_freq(obj)
    obj.query_operation_complete_status();
    response = str2double(obj.query("FREQ?"));
end

function response = read_freq_bin(obj)
    response = obj.query("FREQ:BIN?");
end

function response = read_mfg_sn(obj)
    response = obj.query("MFG:SERNUM?");
end

function response = read_mfg_min_freq(obj)
    obj.query_operation_complete_status(); response =
    str2double(obj.query("MFG:FREQ:MIN?"));
end

function response = read_mfg_freq_step(obj)
    response = str2double(obj.query("MFG:FREQ:STEP?"));
end

function response = read_mfg_default_att(obj)
    response = str2double(obj.query("MFG:ATTEN:DEFAULT?"));
end

function response = read_system_error_query(obj)
    response = obj.query("SYST:ERR?");
end
```



```
function response = read_system_fault_log_query(obj)
    response = obj.query("SYST:FAULT:LOG?");
end

function response = query_operation_complete_status(obj)
    response = logical(str2num(obj.query("*OPC?")));
end

function response = query_status_byte(obj)
    response = obj.query("*STB?");
end

function response = idn(obj)
    response = obj.query("*IDN?");
end

function reset_device(obj)
    obj.write("*RST");
end

end
end
```
