# Tensormeter RTM2 TCP Commands

## 1 Table of Content

2 General Description ..... 2
2.1 Structure of a TCP command ..... 2
3 Commands ..... 3
3.1 All Data [alld] ..... 3
3.2 New Data [newd] ..... 5
3.3 Select Channels [selc] ..... 6
3.4 Clear Data [cldt] ..... 7
3.5 Averaging Time [avgt] ..... 8
3.6 Lockin Frequency [lfrq] ..... 9
3.7 Voltage/Current Setpoints [vodc, cudc, vamp, camp] and Ramps ..... 10
3.8 Voltage/Current Protection [vpro, ipro] ..... 11
3.9 Measurement Ranges [virg, vorg, crng, sres] ..... 12
3.10 Range Increment/Decrement [viru, vird, voru, vord, crup, crdn, srup, srdn] ..... 13
3.11 Switch Task [swit] ..... 14
3.12 Analysis & Multisample Modes [amod, mod?, mult] ..... 16
3.13 Control & Waveform Modes [cmod, wfmd] ..... 17
3.14 Pulse and Arbitrary Waveform Parameters [puar] ..... 18
3.15 Software Triggers for Demodulation and Pulses [trig, puls] ..... 20
3.16 Measurement count [meas] ..... 21
3.17 Digital I/O setup [dio0, dio1] ..... 22
3.18 SNS Preamp Mode & Coax Port Mode [snsa, coax] ..... 23
3.19 Reference MUX, Lock Mode and Phase Shift [refm, phlk, phsh] ..... 24
3.20 Get-All-Server-Settings [gass] ..... 25

# 2 General Description

The whole data communication works via TCP/IP protocol.

The byte order is Big Endian.

The data are sent as **binary string** using the ASCII table and the extended ASCII table [Code Page 1252 Windows Latin 1 (ANSI)].

All data corresponding to physical quantities are in SI units without prefix.

## 2.1 Structure of a TCP command

The commands are always concatenated strings of the following form:

[Length]+[Command]+[Data]

[Length] - 4 bytes, representing the I32 number of the byte length of [Command] + [Data]

[Command] - 4 bytes, the alphabetical command string

[Data] - Could represent anything like: no data, integers, doubles, strings, arrays…

| | [Length] | [Command] | [Data] |
|---|---|---|---|
| **Number of bytes** | 4 Byte | 4 Byte | Zero to Many Bytes |
| **Data Type** | Integer (I32) | string | Command Dependent |
| **Example** | 12 | vamp | 1.243 (double) |
| **Binary form (hex)** | 00 00 00 0C | 76 61 6D 70 | 3F F3 E3 53 F7 CE D9 17 |
| **Concatenated (hex)** | 00 00 00 0C 76 61 6D 70 3F F3 E3 53 F7 CE D9 17 | | |
| **Binary form (string)** | □□□□vamp?óãS÷ÎÙ□ ( □ -> unprintable characters ) | | |

For better understanding, the binary numbers in the following command examples are given in the hexadecimal representation to avoid unprintable characters.

# 3 Commands

## 3.1 All Data [alld]

If the Server receives the all data command, it will send the whole device side data array to the client. The device side array has a maximum length of 8192 data rows. Older data will be lost. To overcome this limitation, use the “New Data” command (next section).

Command: [alld]

Data type: two dimensional array of double

Data unit: column dependent (see “Column & Value” section below)

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 61 6c 6c 64 | | 00 00 00 04 61 6c 6c 64 |

TCP feedback from server:

For example, the server sends an array with two rows and four columns, like this  →



| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 4C | 61 6c 6c 64 | * | 00 00 00 4C 61 6c 6c 64 * |

* 00 00 00 02 00 00 00 04 3F F0 00 00 00 00 00 00 40 00 00 00 00 00 00 00 40 08 00 00 00 00 00 00 40 10 00 00 00 00 00 00 40 14 00 00 00 00 00 00 40 18 00 00 00 00 00 00 40 1C 00 00 00 00 00 00 40 20 00 00 00 00 00 00

The whole TCP answer converted in hexadecimal string and for explanation with some separators |

00 00 00 4C | 61 6C 6C 64 | 00 00 00 02 | 00 00 00 04 | 3F F0 00 00 00 00 00 00 | 40 00 00 00 00 00 00 00 | 40 08 00 00 00 00 00 00 | 40 10 00 00 00 00 00 00 | 40 14 00 00 00 00 00 00 | 40 18 00 00 00 00 00 00 | 40 1C 00 00 00 00 00 00 | 40 20 00 00 00 00 00 00

| Byte | Commend | Data type |
|---|---|---|
| 1-4: | Represent the number of bytes (send as binary) in the answer starting from byte fife up to the end, in this case the number is 76 byte | int (I32) |
| 5-8: | Represent the command string with 4 characters (compare ASCII table) → alld | char |
| 9-12: | Represent the number of rows in the array | int (I32) |
| 13-16 | Represent the number of columns in the array | int (I32) |
| 17-48 | Represent the first row in the array (4 values with 8 bytes each) | double |
| 49-80 | Represent the second row in the array (4 values with 8 bytes each) | double |

The Values in the columns are the following:

| Column | Value |
|---|---|
| 0 | Time (time zone-independent number of seconds that have elapsed since 00:00 [midnight], Friday, January 1, 1904, Universal Time [01-01-1904 00:00:00]. ) |
| 1 | Input Voltage DC (V) |
| 2 | Current DC (A) |
| 3 | Output Voltage DC (V) |
| 4 | Resistance 2W DC (Ohm) |
| 5 | Input Voltage Ampl (V) |
| 6 | Current Ampl (A) |
| 7 | Output Voltage Ampl (V) |
| 8 | Impedance 2W AC (Ohm) |
| 9 | Res A, DC (Ohm) |
| 10 | Res A, 1st Re (Ohm) |
| 11 | Res A, 1st Im (Ohm) |
| 12 | Res A, 2nd Re (Ohm) |
| 13 | Res A, 2nd Im (Ohm) |
| 14 | Res A, 3rd Re (Ohm) |
| 15 | Res A, 3rd Im (Ohm) |
| 16-22 | Same as 9-15, but for „Res B“ instead |
| 23 | Switch Status (U32) |
| 24 | Lock-in Frequency (Hz) |
| 25 | Voltage DC Setpoint (V) |
| 26 | Current DC Setpoint (A) |
| 27 | Current DC Setpoint (A) |
| 28 | Current Ampl Setpoint (A) |
| 29 | Voltage Protection (V) |
| 30 | Current Protection (A) |
| 31 | Input Voltage Peak Range Fill |
| 32 | Current Peak Range Fill |
| 33 | Output Voltage Peak Range Fill |
| 34 | Reference Voltage Peak Range Fill |
| 35 | Voltage Input Range (V) |
| 36 | Voltage Output Range (V) |
| 37 | Current Range (A) |
| 38 | Series Resistance (Ohm) |
| 39 | Sampling Duration (s) |
| 40 | Lock Quality |
| 41 | Analysis & Multisample Mode (U16) |
| 42 | Digital I/O Port 0 (V) |
| 43 | Digital I/O Port 1 (V) |

## 3.2 New Data [newd]

The server sends only the data not yet sent. That means only the last rows of the data array will be sent. Sending this command at regular or irregular intervals guarantees a lossless stream of data to the client. Data rows can still be lost in the device if more than 8192 data rows are acquired by the device between two subsequent “New Data” calls.

Command: [newd]

Data type: two dimensional array of double

Data unit: column dependent (see All Data [alld] command)

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 6E 65 77 64 | | 00 00 00 04 6E 65 77 64 |

TCP feedback from server:

The format is equal to the [alld] command, but the response includes only those data lines which were acquired since the last New Data command.

## 3.3 Select Channels [selc]

With the [selc] command you can select the channels (columns of the data array) you will receive from the New Data command.

Command: [selc]

Data type: one dimensional array of integers (I32)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 14 | 73 65 6C 63 | 00 00 00 03 00 00 00 03 00 00 00 00 00 00 00 02 | 00 00 00 14 73 65 6C 63 00 00 00 03 00 00 00 03 00 00 00 00 00 00 00 02 |

Example Data = 


The whole TCP command converted in hexadecimal string:

00 00 00 14 73 65 6C 63 00 00 00 03 00 00 00 03 00 00 00 00 00 00 00 02

| Byte | Commend | Data type |
|---|---|---|
| 1-4: | Represent the number of bytes (send as binary) starting from byte fife up to the end, in this case the number is 20 byte | int (I32) |
| 5-8: | Represent the command string with 4 characters (compare ASCII table) → selc | char |
| 9-12: | Represent the number of elements in the array | int (I32) |
| 13-16 | Represent the index of the first desired data column | int (I32) |
| 17-20 | Represent the index of the second desired data column | int (I32) |
| 21-24 | Represent the index of the third desired data column | int (I32) |

The server sends a feedback in the same format. If a value is outside the limits, a coerced value is returned.

This command is only used from the client side. When a client connects to the server, the server provides all data channels in ascending order by default. Using the Select Channels command, the client can define an array with any number of data column it likes, in arbitrary order.

## 3.4 Clear Data [cldt]

Delete all data (data array) at server side.

Command: [cldt]

Data type: no data

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 63 6C 64 74 | | 00 00 00 04 63 6C 64 74 |

The server sends a feedback in the same format.

## 3.5 Averaging Time [avgt]

Set the averaging time or sampling period. If multiple Switch States are defined, then switching will always happen synchronously between adjacent sampling periods. The value is sent in seconds.

Command: [avgt]

Data type: double

Data unit: Seconds [s]

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 61 76 67 74 | 3F E0 00 00 00 00 00 00 | 00 00 00 08 6E 65 77 64 3F E0 00 00 00 00 00 00 |

Example Data = 0.5

The server sends a feedback in the same format. If the value is outside the limits, a coerced value is returned.

## 3.6 Lockin Frequency [lfrq]

Set the lock-in frequency in Hz.

Command: [lfrq]

Data type: double

Data unit: Hertz [Hz]

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 6C 66 72 71 | 40 36 80 00 00 00 00 00 | 00 00 00 0C 6C 66 72 71 40 36 80 00 00 00 00 00 |

Example Data = 22.5

The server sends a feedback in the same format. If the value is outside the limits, a coerced value is returned.

This command can be used bidirectionally. If a client is connected, the client can also listen for this command to update the client user interface when the value was changed at the server side.

## 3.7 Voltage/Current Setpoints [vodc, cudc, vamp, camp] and Ramps

There are four setpoints that define the desired output. Each one of these commands sets one of the setpoints:

vodc Voltage Amplitude Setpoint [A]

cudc Current DC Setpoint [A]

vamp Voltage Amplitude Setpoint [V]

camp Current Amplitude Setpoint [A]

Optionally, one can define a ramp time, during which the new setpoint is linearly approached.

The example below uses the “Voltage Amplitude” setting.

Command: [vamp]

Data type: double, (double)

Data unit: Voltage [V], (Ramp Time [s])

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 76 61 6D 70 | 40 1D 4B C6 A7 EF 9D B2 | 00 00 00 0C 76 61 6D 70 40 1D 4B C6 A7 EF 9D B2 |

Example Data = 7.324

If Data is only a single double number, like in the example, no ramp will be applied and the new setpoint will be adopted straight away. If Data contains a second double number, this will be applied as the ramp time.

The server sends live feedback in the same format (without ramp time). If the value is outside the limits, a coerced value is returned.

These commands can be used bidirectionally. If a client is connected, the client can also listen for these command to update the client user interface when the value was changed at the server side.

## 3.8 Voltage/Current Protection [vpro, ipro]

Set the limiting voltage and current protection level.

vpro Voltage Protection [V]

ipro Current Protection [A]

The example is based on the “Voltage Protection” setting.

Command: [vpro]

Data type: double

Data unit: Voltage [V]

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 76 70 72 6F | 40 1D 4B C6 A7 EF 9D B2 | 00 00 00 0C 76 70 72 6F 40 1D 4B C6 A7 EF 9D B2 |

Example Data = 7.324

The server sends a feedback in the same format. If the value is outside the limits, a coerced value is returned.

This command can be used bidirectionally. If a client is connected, the client can also listen for this command to update the client user interface when the value was changed at the server side.

## 3.9 Measurement Ranges [virg, vorg, crng, sres]

There are 4 different measurement and control ranges on the Tensormeter RTM2, namely the Input and Output Voltage ranges, the Current sense range and the Series Resistor setting. Each of these commands sets one of these ranges:

virg Voltage Input Range [V]

vorg Voltage Output Range [V]

crng Current Range [A]

sres Series Resistor [Ohm]

Every positive value is possible. The actually chosen range will be the next largest range that accomodates that specified value. In the case of the “Series Resistor” setting, the actually chosen setting will be the one that is closest (on a log scale) to the specified value. If you send a zero (0.0) or negative value, the device will enter the Auto-Range mode. While in the Auto-Range mode, the server sends range updates with a negative sign, e.g. “-0.5”. To return to manual range mode, just send a positive value for the range, (e.g. “0.8”). The example below uses the “Voltage Input Range” setting.

Command: [virg]

Data type: double

Data unit: Voltage [V]

Whole TCP command for [virg]:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 76 69 72 67 | 40 00 00 00 00 00 00 00 | 00 00 00 0C 76 69 72 67 40 00 00 00 00 00 00 00 |

Example Data = 2.0

The server sends a feedback in the same format.

These commands can be used bidirectionally. If a client is connected, the client can also listen for these commands to update the client user interface when the value was changed at the server side.

## 3.10 Range Increment/Decrement [viru, vird, voru, vord, crup, crdn, srup, srdn]

Increments or Decrements a specific measurement range setting and disables Auto-Range mode for this specific range. To re-enable Auto-Range mode, use the explicit measurement range commands (Section 3.9).

viru Voltage Input Range Up

vird Voltage Input Range Down

voru Voltage Output Range Up

vord Voltage Output Range Down

crup Current Range Up

crdn Current Range Down

srup Series Resistor Up

srdn Series Resistor Down

The following example uses the “Voltage Output Range Up” command.

Commands: [voru]

Data type: no data

Data unit: no unit

Whole TCP command for [voru]:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 76 6F 72 75 | | 00 00 00 04 76 6F 72 75 |

The server sends a feedback in the form of the corresponding range command, in this example, the answer would be the new “Voltage Output Range”, as explained in Section 3.9).

## 3.11 Switch Task [swit]

Switch task sends a one dimensional array of U32 values. Each value represents a switch state of the Tensormeter RTM2.

Command: [swit]

Data type: one dimensional array of uint (U32)

Data unit: no unit

Whole TCP command:

For example, sending an array with two values, like this →


| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 10 | 73 77 69 74 | 00 00 00 02 00 00 00 00 00 00 00 01 | 00 00 00 10 73 77 69 74 00 00 00 02 00 00 00 00 00 00 00 01 |

The whole TCP command converted in hexadecimal string:

00 00 00 10 73 77 69 74 00 00 00 02 00 00 00 00 00 00 00 01

| Byte | Commend | Data type |
|---|---|---|
| 1-4: | Represent the number of bytes (send as binary) starting from byte fife up to the end, in this case the number is 16 byte | int (I32) |
| 5-8: | Represent the command string with 4 characters (compare ASCII table) → swit | char |
| 9-12: | Represent the number of elements in the array | int (I32) |
| 13-16 | Represent the first state of the array (32 switch bits = 4 bytes) | uint (U32) |
| 17-20 | Represent the second state of the array (32 switch bits = 4 bytes) | uint (U32) |

The server sends a feedback in the same format.

This command can be used bidirectionally. If a client is connected, the client can also listen for this command to update the client user interface when the value was changed at the server side.

**The switch state is calculated in the following way: (next page)**

Each switch state is represented by a 32 bit unsigned integer value. Each bit represents a possible connection between an internal analog function and a BNC connector at the front panel of the Tensormeter RTM2.

**Example 1:** If the Bit 10 is high, the BNC port 3 connector is connected to the internal function DRV+ of the Tensormeter.

| | **BNC 1** | **BNC 2** | **BNC 3** | **BNC 4** | **BNC 5** | **BNC 6** | **BNC 7** | **BNC 8** |
|---|---|---|---|---|---|---|---|---|
| **DRV-** | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
| **DRV+** | Bit 8 | Bit 9 | Bit 10 | Bit 11 | Bit 12 | Bit 13 | Bit 14 | Bit 15 |
| **SNS-** | Bit 16 | Bit 17 | Bit 18 | Bit 19 | Bit 20 | Bit 21 | Bit 22 | Bit 23 |
| **SNS+** | Bit 24 | Bit 25 | Bit 26 | Bit 27 | Bit 28 | Bit 29 | Bit 30 | Bit 31 |

If only the Bit 10 is high, and all other Bits are low, the Switch State value (U32) will be $2^{10}$ = 1024 (decimal notation).

| Bit 31 - 24 (SNS+) | Bit 23 - 16 (SNS-) | Bit 15 - 8 (DRV+) | Bit 7 - 0 (DRV-) |
|---|---|---|---|
| 0000 0000 | 0000 0000 | 0000 0100 | 0000 0000 |
| 0000 0000 0000 0000 0000 0100 0000 0000 (binary) | | | |
| 00 00 04 00 ( hex ) | | | |
| 1024 ( dec ) | | | |

**Example 2:** If several Bits, e.g. Bits 0, 10, 17 and 27 are high, the Switch State value (U32) is the sum of the value of the individual bits

| | **BNC 1** | **BNC 2** | **BNC 3** | **BNC 4** | **BNC 5** | **BNC 6** | **BNC 7** | **BNC 8** |
|---|---|---|---|---|---|---|---|---|
| **DRV-** | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
| **DRV+** | Bit 8 | Bit 9 | Bit 10 | Bit 11 | Bit 12 | Bit 13 | Bit 14 | Bit 15 |
| **SNS-** | Bit 16 | Bit 17 | Bit 18 | Bit 19 | Bit 20 | Bit 21 | Bit 22 | Bit 23 |
| **SNS+** | Bit 24 | Bit 25 | Bit 26 | Bit 27 | Bit 28 | Bit 29 | Bit 30 | Bit 31 |

In this example the decimal Switch State value is $2^0 + 2^{10} + 2^{17} + 2^{27}$ = 1 + 1024 + 131,072 + 134,217,728 = 134,349,825

| Bit 31 - 24 (SNS+) | Bit 23 - 16 (SNS-) | Bit 15 - 8 (DRV+) | Bit 7 - 0 (DRV-) |
|---|---|---|---|
| 0000 1000 | 0000 0010 | 0000 0100 | 0000 0001 |
| 0000 1000 0000 0010 0000 0100 0000 0001 (binary) | | | |
| 08 02 04 01 ( hex ) | | | |
| 134,349,825 ( dec ) | | | |

## 3.12 Analysis & Multisample Modes [amod, mod?, mult]

The Analysis Modes and Multisample Modes control how the device handles Switch Tasks and what is being calculated and outputted via the data array. Each command affects a specific setting.

amod Analysis Mode requested

mod? Analysis Mode actual (only used by the server)

mult Multisample Mode

Each of these commands use a single Byte of data representing an 8-Bit unsigned integer (U8). The numeric value corresponds to the mode. Possible Analysis and Multisample Modes are:

| Analysis Modes | |
|---|---|
| 0 | Auto |
| 1 | Kelvin |
| 2 | Zero-Offset Hall |
| 3 | Van-der-Pauw |
| 4 | Ratiometric |
| 5 | Differential |

| Multisample Modes | |
|---|---|
| 0 | Off |
| 1 | Interleave |
| 2 | Differential |
| 3 | Ratiometric |

The following example sets the requested Analysis Mode to “Zero-Offset Hall”:

Command: [amod]

Data type: unsigned integer (U8)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 05 | 61 6D 6F 64 | 02 | 00 00 00 05 61 6D 6F 64 02 |

Example Data = 2 (Sets the Zero-Offset-Hall mode)

Whenever clients change the Analysis or Multisample Modes, the server sends three answers, one each related to [amod], [mod?] and [multi]. All three answers follow the same format as shown in the example.

**Note:** While the “Requested Analysis Mode” is set to “Auto”, the “Actual Analysis Mode” depends on the Switch Task (Section 3.11). A change of the Switch Task can thus lead to a change in the “Actual Analysis Mode” that is not automatically sent by the server via the [mod?] command. The information about the “Actual Analysis Mode” can be either found in the data array column 41, or can be asked for by sending an [amod -> “Auto”] command.

## 3.13 Control & Waveform Modes [cmod, wfmd]

These modes control the output from the DRV analog functions. The meaning of the integer Data is shown below:

| Control Modes | | Waveform Modes | |
|---|---|---|---|
| 0 | Direct Voltage Output | 0 | Continuous Sine Wave |
| 1 | Feedback Voltage/Current Output | 1 | Pulse Train |
| | | 2 | Arbitrary Waveform |

The example below sets the “Feedback” operation as “Control Mode”.

Set the value for the control mode.

Command: [cmod]

Data type: unsigned integer (U8)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 05 | 63 6D 6F 64 | 01 | 00 00 00 06 63 6D 6F 64 01 |

Example Data = 1 (Sets the “Feedback Voltage/Current Output” mode)

The server sends a feedback in the same format.

This command can be used bidirectionally. If a client is connected, the client can also listen for this command to update the client user interface when the value was changed at the server side.

## 3.14 Pulse and Arbitrary Waveform Parameters [puar]

The [puar] command sets the parameters of the pulse trains or arbitrary waveforms to be outputted. The interpretation depends on the currently set “Waveform Mode” (Section 3.13).

**Example 1 for “Pulse Train” mode definition:**

In “Pulse Train” mode, the Data for [puar] should be a 1-d array consisting of 6 double numbers, i.e. 48 Byte in total:

1. Period Time [s] (double, 8 Byte)
2. “On” Time [s] (double, 8 Byte)
3. “On” Voltage [V] (double, 8 Byte)
4. “Off” Voltage [V] (double, 8 Byte)
5. Duration [number of periods] (double, 8 Byte)
6. Starting phase [fraction of periods] (double, 8 Byte)

Define five 10-µs-pulses, 100 µs apart, 1 V when “on”, 0 V when “off”, starting with the “on” phase (i.e. the start phase is “0.0 periods”). The Data will be (in decimal notation): 100e-6, 10e-6, 1, 0, 5, 0

| | Length | Command | Data (example) | Whole TCP command |
|---|---|---|---|---|
| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
| **HEX** | 00 00 00 38 | 70 75 61 72 | * [52 bytes total] | 00 00 00 38 70 75 61 72 * |

The whole TCP answer converted in hexadecimal string and for explanation with some separators |

00 00 00 38 | 70 75 61 72 | 00 00 00 06 | 3F 1A 36 E2 EB 1C 43 2D | 3E E4 F8 B5 88 E3 68 F1 | 3F F0 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 | 40 14 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00

| Byte | Commend | Data type |
|---|---|---|
| 1-4: | Represent the number of bytes (send as binary) in the answer starting from byte 5 up to the end, in this case the number is 56 byte | int (I32) |
| 5-8: | Represent the command string with 4 characters (compare ASCII table)→ puar | char |
| 9-12 | Represent the number of elements in the array | int (I32) |
| 13-20 | Represent the first element in the array | double |
| 21-28 | Represent the second element in the array | double |
| 29….. | and so on | … |

The server sends a feedback in the same format.

**Example 2 for “Arbitrary Waveform” mode definition:**

In “Arbitrary Waveform” mode, the Data for [puar] should be an even number of double numbers, i.e. at least 2 numbers, where each pair of double numbers defines one step of the arbitrary waveform:

1. Point Voltage [V] (double, 8 Byte)
2. Hold Time [s] (double, 8 Byte)

Define two steps, first output 1 V for 100 µs, then output -0.1 V for 2 ms. The Data will be (in decimal notation): 1, 100e-6, -0.1, 2e-3

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 28 | 70 75 61 72 | * [36 bytes total] | 00 00 00 28 70 75 61 72 * |

The whole TCP answer converted in hexadecimal string and for explanation with some separators |

00 00 00 28 | 70 75 61 72 | 00 00 00 04 | 3F F0 00 00 00 00 00 00 | 3F 1A 36 E2 EB 1C 43 2D | BF B9 99 99 99 99 99 9A | 3F 60 62 4D D2 F1 A9 FC

The server sends a feedback in the same format.

## 3.15 Software Triggers for Demodulation and Pulses [trig, puls]

Commands: [trig] Begin a new demodulation phase instantly
[puls] Begin pulse/arbitrary waveform output instantly
Data type: no data
Data unit: no unit

Whole TCP command for [trig]:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 74 72 69 67 | | 00 00 00 04 74 72 69 67 |

Whole TCP command for [puls]:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 70 75 6C 73 | | 00 00 00 04 70 75 6C 73 |

The server sends a feedback in the same format.

## 3.16 Measurement count [meas]

Sets the number of data points that will be still written into the device side data buffer before suspending any further storage of measurement results. As measurements are being done, the number will decrement to 0. Updates are automatically posted to the client. A value of -1 will never decay, indicating continuous measurements. This is also the default state of the device.

Commands: [meas]

Data type: signed integer (I32)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 08 | 6D 65 61 73 | 00 00 00 14 | 00 00 00 08 6D 65 61 73 00 00 00 14 |

Example Data = 20

The server sends a feedback in the same format.

This command can be used bidirectionally. If a client is connected, the client can also listen for this command to update the client user interface when the value was changed at the server side.

**Note:** A change of the desired measurement count will not trigger a demodulation. Rather, the device will continue demodulation at a constant rate and begin storing measurement results when they appear.

**Note:** Be careful about when to retrieve measurement data, e.g. via the “newd” command. Any call for data array entries will execute immediately and not wait for the “Measurement count” to reach zero. However, there is no harm in requesting data well after the measurement phase has completed, as there is no new data that could potentially overwrite the stored data points in the device data buffer.

## 3.17 Digital I/O setup [dio0, dio1]

Sets either Digital I/O port 0 or 1 into a specified operation mode. There are a total of 9 Bytes of parameters: an 8-bit unsigned integer sets the basic operation mode and an additional double number represents a voltage value that manipulates some of these modes.

| DIO Mode (decimal): | | Voltage [V]: |
|---|---|---|
| 0 | Transparent Input | |
| 1 | Hardware Trigger Input Demodulation | |
| 2 | Hardware Trigger Input Pulse | |
| 3 | 20-bit Counter Input | |
| 4 | Interlock signal Input for DRV | |
| 128 | Transparent Output | LO output (<1.65), HI output (>= 1.65) |
| 129 | Sync Output Demodulation | |
| 130 | Sync Output Pulse | |
| 131 | Delta-Sigma DAC Output (unfiltered) | Analog output voltage [V]: 0 … 3.3 |
| 132 | DRV Limiting Indicator | |

As an example, we set DIO Port 0 into the DAC mode with an output of 0.5 V.

Whole TCP command:

| | Length | Command | Data (example) | Whole TCP command |
|---|---|---|---|---|
| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
| **HEX** | 00 00 00 0D | 64 69 6F 30 | 83 \| 3F E0 00 00 00 00 00 00 | 00 00 00 0D 64 69 6F 30 83 \| 3F E0 00 00 00 00 00 00 |

The server sends a feedback in the same format.

## 3.18 SNS Preamp Mode & Coax Port Mode [snsa, coax]

These modes control the internal device configuration around the front side analog ports. The meaning of the integer Data is shown below:

snsa SNS Preamp type

coax Coax Shell Mode

SNS Preamp Modes

0 BJT Preamp

1 FET Preamp

Coax Shell Modes

0 Ground Shells

1 Active Guard all active ports

2 Guard exclusive SNS ports only

3 Guard exclusive DRV ports only

**Example:** Enable the FET Preamp

Command: [snsa]

Data type: unsigned integer (U8)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 05 | 73 6E 73 61 | 01 | 00 00 00 06 73 6E 73 61 01 |

The server sends a feedback in the same format.

## 3.19 Reference MUX, Lock Mode and Phase Shift [refm, phlk, phsh]

These modes control the Reference Lock circuitry. The meaning of the integer Data is shown below:

refm Reference Multiplexer

phlk Phase Lock Source

Reference Multiplexer

| | |
|---|---|
| 0 | Off |
| 1-8 | Stay at Port 1-8 |
| 9 | Follow SNS+ |
| 10 | Follow SNS- |
| 13 | Follow DRV+ |
| 14 | Follow DRV- |

Phase Lock Source

| | |
|---|---|
| 0 | Internal Clock |
| 1 | Reference Multiplexer |

The [phsh] command specifies a phase shift of the DRV wave with respect to the Reference source. The Data for this command is a double number corresponding to the fraction of cycles to offset the phase.

**Example1:** Enable the phase locking onto an external reference source

Command: [phlk]

Data type: unsigned integer (U8)

Data unit: no unit

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 05 | 70 68 6C 6B | 01 | 00 00 00 06 70 68 6C 6B 01 |

**Example:** Specify a phase shift of 90° (π/2) between reference and DRV, i.e. “0.25” cycles.

Command: [phsh]

Data type: double

Data unit: cycles/periods

Whole TCP command:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 0C | 70 68 73 68 | 3F D0 00 00 00 00 00 00 | 00 00 00 0C 70 68 73 68 3F D0 00 00 00 00 00 00 |

The server sends a feedback in the same format.

## 3.20 Get-All-Server-Settings [gass]

[gass] command request all device settings.

Command: [gass]

Data type: no data

Data unit: no unit

Whole TCP command for [gass]:

| | | The 4 Byte length corresponds to the sum of all Bytes from Command + Data | | |
|---|---|---|---|---|
| | **Length** | **Command** | **Data (example)** | **Whole TCP command** |
| **HEX** | 00 00 00 04 | 67 61 73 73 | | 00 00 00 04 67 61 73 73 |

The server sends all current settings using the respective data formats associated with each command, as discussed in the previous sections.