proceq

Platform Remote Control Package

Operating Instructions

Swiss Precision since 1954

Table of Contents

1. Safety and Liability

1.1 Safety and Usage Precautions

This manual contains important information on the safety, use and maintenance of the Platform remote control package software. Read through the manual carefully before the first use of the software. Keep the manual in a safe place for future reference.

1.2 Liability and the “Software License Agreement”

Our “General Terms and Conditions of Sale and Delivery” apply in all cases.
All information contained in this documentation is presented in good faith and believed to be correct. Proceq SA makes no warranties and excludes all liability as to the completeness and/or accuracy of the information.

The subject of the license is the software that Proceq dispatches to the customer by electronic means, in the version as sent and in the form as received by the customer. Proceq accepts no responsibility for any properties of the software, either general or in terms of its suitability for any particular purpose. The customer carries the risk that the software may not meet his expectations in terms of results or performance. Proceq shall also not be liable for any damage suffered by the customer or third party irrespective of whether the damage is the direct or indirect consequence of installing or using the software or is in any way connected with installation or use of the software. In particular, Proceq shall not be liable for loss of earnings, profit or savings, or for loss of or damage to software or data. This applies even if Proceq is expressly made aware of such risks. Exclusion of liability applies in particular to all direct, indirect or consequential damage that may arise to the customer because the software fails to function properly or at all for whatever reason.

2. Overview

2.1 Scope / Intended Use

This document describes the software part of the Platform remote control package including the syntax of several Proceq remote commands.
The target audience is Software Engineers who want to embed Platform indicating devices (e.g. Equotip 550, Profometer PM-6, and Pundit PL-2) into their own Applications.

2.2 Purpose

Some customers want to use their Proceq device in a (semi-) automated production.
Some want to be able to get the measurements into their own Software and databases to be further processed or stored without the need of Proceq Link Software and overhead of unneeded manual work.
Some might want to remotely setup their device into a specific, well defined measurement mode with one click.
Some have other needs or ideas about how their Proceq device has to be embedded into their workflow.

The purpose of the Platform remote control package add-on is to fulfill as many of these customer needs as possible in a most flexible way.

2.3 Solution

The Proceq platform indicating device has two physical communication interfaces to a PC (Ethernet and USB). On each of these interfaces it listens to incoming commands and sends out data accordingly.

The commands are 2 (legacy) or 4 bytes ASCII followed by a well defined number of parameters per command. They are documented in here and can be used by Software Engineers in their own applications.

To make life easy for programmers under MS Windows, Proceq provides an interface header file and a DLL they can use (aka PqRemote library). This DLL provides functions for device connection, sending commands and receiving data. In addition, application examples show how to use the PqRemote library.

Programmers under Linux and other Operating Systems will have to open the interface and connect to the device by themselves. They must know and use the low level escape and acknowledge sequences together with the ASCII commands. This is a bit of work, but they still can use their Proceq platform device remotely and in an automated environment.

2.4 Requirements, prerequisites, limitations

To avoid problems during remote connection please take care of the following:

- Make sure the device is powered on, connected to the PC, and the application is running.
- If possible, have the device powered with the AC adapter.
- Disable power management of the device: Read the operating instruction of your device in case you need help regarding this.
- The Equotip 540 versions don't support PqRemote. Upgrade to Equotip 550 if you need PqRemote.
- Make sure, the measurement series are closed either automatically or remotely after a certain amount of measurements (see chapter 5.3 `SLEN`).
- Make sure you use the latest device firmware and the latest Platform Remote Control Package (see chapter 5.1.3 Missing commands, missing functionality)

2.5 Abbreviations

Term	Description
HW	Hardware
SW	Software
USB	Universal serial bus
PC	Personal Computer
DLL	Dynamic Link Library

3. Parts of PqRemote

3.1 File and directory overview

Please find below an overview of the files in the Platform Remote Control Package `PqRemote_V_x_y_z.zip`.
This package should be downloadable via Link tool and the Proceq homepage. It may be part of the software that comes along with the Proceq product you bought.

Contents of "PqRemote" Platform Remote Control Package	**Description**
Documentation	Folder with Package Documentation
Platform Remote Control Package_OI_E.pdf	This document
SourceCode	Folder with application samples using PqRemote
PqRemoteDemoDLL	Sample which uses PqRemote.dll, PqRemoteX64.dll
res	Folder containing resource files
.	resource files
PqRemoteDemoDLL_VS9_PC.vcproj	Visual Studio 2008 project file
PqRemoteDemoDLL_VS9_PC. sln	Visual Studio 2008 solution file
PqRemoteDemoDLL_VS14_PC.vcxproj	Visual Studio 2015 project file
PqRemoteDemoDLL_VS14_PC. sln	Visual Studio 2015 solution file
PqRemoteDemoDLLDlg.cpp	The source file using the PqRemote library with PqRemote.dll (PqRemoteX64.dll for x64)
PqRemoteDemoDLL.cpp	The main source file. Defines the application and starts the dialog
.	Other C++ source and include files.
PqRemoteDemolib	Sample which uses PqRemoteX64.dll, PqRemote.dll together with the import library
PqRemoteDemolibDlg.cpp	The source file using the PqRemote library with PqRemote.lib import library and PqRemote.dll
.	Similar files as in PqRemoteDemoDLL
Binaries	Folder with the PqRemote libraries plus executable samples
PqRemote.h	The PqRemote interface header file
PqRemote.dll	The 32 bit PqRemote dynamic library
PqRemote.lib	The 32 bit PqRemote import library
PqRemoteX64.dll	The 64 bit PqRemote dynamic library
PqRemoteX64.lib	The 64 bit PqRemote import library
DemoRemote.exe	Sample which uses SCSH and KEYB to remotely control the device
PqRemoteDemoExcel.xlsm	Sample which shows how to use PqRemote in Excel with 10 Measurements max.
PqRemoteDemoExcel64.xlsm	Same as above but for x64 with newer Excel
PqRemoteDemoExcel64_2.xlsm	2nd version for x64 with Excel 2007and newer.
PqRemoteDemoExcel500.xlsm	Sample which shows how to use PqRemote in Excel with 500 Measurements max.
PqRemoteDemolib_VS14_PC.exe	The resulting .exe of PqRemoteDemolib built with Studio 2015 for 32 bit (Win32)
PqRemoteDemoDLL _VS9_PC.exe	The resulting .exe of PqRemoteDemoDLL built with Studio 2008 for 32 bit (Win32)
PqRemoteDemolib_VS9_PCX64.exe	The resulting .exe of PqRemoteDemolib built with Studio 2008 for 64 bit (x64)
PqRemoteDemoDLL_VS14_PCX64.exe	The resulting .exe of PqRemoteDemoDLL built with Studio 2015 for 64 bit (x64)

3.2 The PqRemote library

3.2.1 PqRemote.h

The interface header file describing the functions exported by PqRemote.dll and PqRemoteX64.dll

If you are programming in C or C++, you should be able to include this header file into your application and use either the functions as declared in the interface header (see PqRemoteDemoLib) or the defined pointers together with GetProcAddress (see PqRemoteDemoDLL).

If you are programming in another language, you can use this file only for documentation purpose and will have to write your own interface file or use the exported methods of the PqRemote.dll in another way. Take a look at the PqRemoteDemoExcel.xlsm and PqRemoteDemoExcel500.xlsm which shows the usage of the DLL in Excel with Visual Basic macros.

3.2.2 PqRemote.dll, PqRemoteX64.dll

This dynamic link library contains the function implementation declared in PqRemote.h. It has to be distributed together with your application. It should be located in the same folder as the application or in the appropriate windows system folder.
PqRemote.dll is for 32 bit applications, PqRemoteX64 is for 64 bit applications. The appropriate dll is compatible with 32 bit / 64 bit applications for Windows XP, Windows Vista, Windows 7, Windows 8.1 and Windows 10. The normal issues running 64 bit applications on 32 bit Windows versions apply, so for 32 bit Windows you should use the 32 bit dll and build a 32 bit application. If you have to write your application for another OS, see [4. Remote commands low level Protocol].

3.2.3 PqRemote.lib, PqRemoteX64.lib

This is an import library describing the exports of the DLL. It is not a static library that can be linked to your application. You still have to distribute the PqRemote.dll or PqRemoteX64.dll together with your application, but this import library simplifies the usage of the DLL if you are programming in C++, see PqRemoteDemoLib as a sample project.

3.2.4 The exported functions

3.2.4.1 Data Types, Specifiers and Modifiers

Please find below the data types, specifiers and modifiers used in exported functions.
NOTE: The specified length of data types equals to the amount of storage required in Microsoft C++.

`char`	An integral type with a size of 1 byte
`short`	An integral type with a size of 2 bytes
`long`	An integral type with a size of 4 bytes
`char *`	A pointer (4 bytes Win32, 8 bytes x64) to a memory area containing data of type char.
`void *`	A pointer (4 bytes Win32, 8 bytes x64) to a memory area containing unspecified data type.
`bool`	An integral type that can have the value true or false. Its size is 1 byte
`signed`	The following data type is signed. If neither signed nor unsigned is specified, the type is assumed signed.
`unsigned`	The following data type is unsigned.
`const`	Here: Specifies that the following variable will not be modified in the function.

3.2.4.2 PQ_CheckCompatibility ( )

```
unsigned long PQ_CheckCompatibility(const unsigned short mainVersion,
                                    const unsigned short sideVersion);
```

Description

Checks if the loaded DLL is compatible with the interface version you use in your application.

The Product Version of DLLs has the format **a.b.c.d** and can be read out manually in file explorer under properties. Where **a** is the main version and **b** is the side version (**c** and **d** can be ignored for our compatibility check). The main and side version numbers (**a.b**) are defined in PqRemote.h, too:

```
#define PQ_REMOTE_INTERFACE_MAIN_VERSION 1
#define PQ_REMOTE_INTERFACE_SIDE_VERSION 0
```

So you can do the check in your application like in the sample code snippet below:

```
unsigned long result=PQ_CheckCompatibility(PQ_REMOTE_INTERFACE_MAIN_VERSION,
                                           PQ_REMOTE_INTERFACE_SIDE_VERSION);
if(result > 1) return error;
```

Parameters

mainVersion [IN]	The main version number of the PqRemote interface (PqRemote.h) you are using.
sideVersion [IN]	The side version number of the PqRemote interface (PqRemote.h) you are using.

Return value

0 =	Everything ok, the version matches.
1 =	The side version of the dll is newer, but the dll is backwards compatible → ok.
2 =	The side version of the dll is older, you should use a new dll.
3 =	The dll main version does not match, do not use this dll with your application.
4 =	An unexpected Software problem occurred.

3.2.4.3 PQ_ConnectDevice ( )

```
unsigned long PQ_ConnectDevice(const unsigned short devType,
                               const unsigned short commPort,
                               const bool           scanPort,
                               const unsigned short portNr,
                               const char          *ipAddress,
                               const unsigned long  waitMS);
```

Description

Connect to a Proceq device of type `devType` using the communication Port (`commPort`).
You receive a device handle > 0 in case of success or 0 in case of failure.

The parameters which are not needed by the chosen `commport` are ignored.
E.g. if you choose 4 (Ethernet), it does not matter what you specify in `scanPort` and `portNr`.

Parameters

`devType`	`[IN]`	Specifies the Proceq device type you want to connection with: 0 = Equotip3; 1 = Piccolo; 2 = SilverSchmidt; 3 = Profoscope; 10 = Platform
`commPort`	`[IN]`	The communication port you want to use for the connection: 1 = Serial(RS232); 2 = USB; 4 = Ethernet NOTE: if you use USB (2) and have more than one device of the appropriate type connected to the PC via USB, the connection will be established with the first device found.
`scanPort`	`[IN]`	False = uses only portNr True = scans ports until a device is found or the maximum port number is reached (255). This parameter is only used when comport is 1 = Serial (RS232).
`portNr`	`[IN]`	The serial COM port number the device is connected to (only RS232).
`ipAddress`	`[IN]`	The IP address of the device as string e.g. "192.168.0.204" (only Ethernet). Socket = 3728 (0xE90)
`waitMS`	`[IN]`	Maximum time in [ms] to wait in this method. NOTE: If you are using Ethernet, the underlying socket (3728) might still be working in case you set `waitMS` < 25000 ms. This means that the responsible thread in the DLL cannot process any other ConnectDevice before the socket responded.

Return value

`0 =`	No device was found or the timeout `waitMS` occurred.
`> 0 =`	A handle to the device that was found. You will need this handle as parameter for any other function which communicates with the device.

3.2.4.4 PQ_DisconnectDevice ( )

```
bool PQ_DisconnectDevice(const unsigned long deviceHandle);
```

Description

Disconnects the device previously connected with PQ_ConnectDevice().

Parameters

`deviceHandle`	`[IN]`	The handle to the device you want to disconnect.

Return value

`false`	The device handle was invalid (no connected device with this handle was found).
`true`	The connection was successfully closed.

3.2.4.5 PQ_SendCMD ( )

```
long PQ_SendCMD(const unsigned long deviceHandle,
                const char *cmd,
                char *reply,
                const unsigned long maxReplyLen,
                const long timeout);
```

Description

Send an ASCII command (`cmd`), see [5. Remote ASCII commands] to the device and receive the `reply`. The return value is the reply length or -1 in case of failure.

With few exceptions, the answer of all commands can be received with the reply buffer of this function immediately.

NOTE: You are responsible for the reply buffer (you have to allocate and free the memory assigned to the buffer). A reply buffer size of 128 bytes should be enough for the reply of all commands except for the command `"DIRS"`.

All commands acknowledge here, but with few commands, additional data must be received using [3.2.4.6 PQ_Read ( )]. These commands are: `"HELP"`, `"SCSH"`, `"GETF"`, `"EMSG"`

Parameters

`deviceHandle`	`[IN]`	The handle to the device you want to send the command to.
`cmd`	`[IN]`	The null terminated ASCII command string including its parameters.
`reply`	`[IN]`	A pointer (memory address) to the buffer where the reply from the device will be stored.
	`[OUT]`	The ASCII reply from the device.
`maxReplyLen`	`[IN]`	The size of the reply buffer in bytes.
`timeout`	`[IN]`	The read and write timeout of the connection in [ms]. The function will return if this timeout elapsed while sending the command or receiving the reply. If you specify -1, the default timeout will be used.

Return value

`-1`	An error occurred (connection, timeout, parameter, wrong `cmd`)
`0`	The cmd was successfully received by the device without a reply in reply buffer.
`N > 0`	The cmd was successfully received and the device answered. The return value is the answer length in bytes.

3.2.4.6 PQ_Read ( )

```
long PQ_Read(const unsigned long deviceHandle,
             void *buffer,
             const unsigned long size,
             const long timeout);
```

Description

Read data from the communication port the device is connected to into the buffer.

The function will return when `size` bytes are read or when the `timeout` occurred.
If the `timeout` occurred, any bytes read before the `timeout` will be in `buffer`.

NOTE: You are responsible for the buffer (you have to allocate and free the memory assigned to the buffer).

Parameters

`deviceHandle`	`[IN]`	The handle to the device you want to read from.
`buffer`	`[IN]`	A pointer (memory address) to the buffer where the read bytes will be stored.
	`[OUT]`	The bytes read from the device.
`size`	`[IN]`	The size of the buffer in bytes.
`timeout`	`[IN]`	The read timeout of the connection in [ms]. The function will return after this timeout elapsed. If you specify 0, the bytes already received by the lower layer communication stack will be returned immediately. If you specify -1, the default timeout will be used.

Return value

`-1`	An error (file exception) occurred.
`0`	No single byte was read before timeout.
`> 0`	Number of bytes read and transferred to buffer

3.3 The Samples

The provided samples show how you could embed and use PqRemote in your application or in Microsoft Excel.
The executables are written in C++ and compiled with Visual Studio 2008 as 32 bit applications.

In the Binaries directory you will find the compiled release version of the applications, the Excel sample and the PqRemote.dll, PqRemoteX64.dll. You can run the samples without the need of compiling them.

3.3.1 PqRemoteDemoDLL

This sample shows a dialog in which you can play around with the exported functions of PqRemote.

Most parameter of the functions can be set in the dialog with some exceptions:

Pq_SendCMD:	The reply buffer size is defined in the source code. The default timeout is taken.
Pq_Read:	The read buffer size is defined in the source code.

In this sample, the dll is loaded with `LoadLibrary()`. In case the dll could be loaded, pointer to the functions of the dll are received with `GetProcAddress()`.
The PqRemote specific code can be found in PqRemoteDemoDLLDlg.cpp

3.3.2 PqRemoteDemoLib

This sample is similar to PqRemoteDemoDLL. But since the import library PqRemote.lib is used during compile and link time, the functions of PqRemote can be called directly (see `PqRemoteDemolibDlg.cpp`).

NOTE: You still need PqRemote.dll or PqRemoteX64.dll for 64 bit together with your Application, but the import library is no longer needed once the .exe is built.

3.3.3 DemoRemote

With this tool you can receive and save screen shots from a connected Platform device. It uses the Remote Command `SCSH`.
In addition, you can enable the keypad with which key commands can be sent to the indicating device with the command `KEYB`.
With the mouse (left mouse button) you can simulate touch events (DemoRemote uses the command `TOUC)`.
In case you press and hold the <ctrl> key, you are in two finger mode: Press and release the left mouse button will set the position of the first finger which will be constant and static. You can now move the mouse, press and hold the left mouse button to set the start position of the second finger which can be moved for as long as the left mouse button is down. Alternatively when <ctrl> is pressed, you can use the mouse-wheel to zoom in and out, but make sure the mouse cursor is not too close to a border of the zoom able area.
With `TOUC` and `KEYB` you should be able to remotely control the Platform device, as proved with DemoRemote.
This sample is not available as source code; you only have the .exe (DemoRemote.exe).
This tool was intentionally programmed for internal use only. You have to use it at your own risk, we don't claim it's bug-free.

Proceq uses this sample together with a laptop and a beamer for demonstrating the Platform applications to a big audience. It's useful to get screen shots for manuals or to see what's displayed in case the LCD of the Platform cannot be read anymore (temperature tests with temperatures below -10°C).

3.3.4 PqRemoteDemoExcel

This sample demonstrates the usage of PqRemote.dll in Excel. It supports Equotip 3 and Equotip 550 for now. To run this sample, you have to enable macros. In Excel 2007: Start→Excel Options→Trust Center→Trust Center Settings→Macro Settings→Enable all macros.

The sample is written in Visual Basic. To view, debug or change the code, you have to enable (show) the developer tab. In Excel 2007: Start→Excel Options→Popular→Show developer tab in the Ribbon. There click on Visual Basic→Module1 and you should see the code.

Please make sure, the PqRemote.dll is in the same folder as the PqRemoteDemoExcel.xlsm so that Excel finds the dll.

NOTE: Should you get a Compile Error similar to “Method or Data Member not found”, the cause is most likely a Microsoft Security Update. There is a patch for it: http://support.microsoft.com/kb/3025036/EN-US → And there just click on “Fix It”. Should the link here not work, you can search for the knowledge base article KB3025036.

4. Remote commands low level Protocol description

4.1 Preamble

If you are using *PqRemote.dll or PqRemoteX64.dll* you don't have to read this subchapter. It describes the low level protocol of the Proceq ASCII remote commands.

This protocol must be followed by programmers who are writing applications for Linux and other Operating Systems beside Microsoft Windows.

There is no USB driver for other OS than Microsoft Windows, so you have to use Ethernet on those Operating Systems.

4.2 General Syntax

All commands from the PC to the device start with the ASCII command character **<DLE>** (0x10, 16 decimal). The device answers always with the character ('**>**').

It follows a two (legacy commands) or four byte ASCII **command code**. It is not case sensitive.

Most of the commands have **parameters**, which are also in ASCII. You can but don't have to separate the command and the first parameter with a space (' '). Parameter containing text can be enclosed in quotation marks (""). Multiple parameters are separated by semicolons ('**;**') which can but don't have to be followed by spaces (' ').

Commands which allow requests and settings use the question mark ('?') as request.

The ASCII command from PC to device is ended with a carriage return ('\r').

If echo mode is on, the device answers to the carriage return with a space (' '), then it executes the command and sends its answer. The answer from the device is terminated with return and linefeed ("\r\n") if echo mode is on, or with carriage return only ('\r') if echo mode is off.

If the command could not be executed (wrong command, wrong parameter or not allowed due to wrong activation code or not logged in), the answer from the device will be a question mark ('?').

ASCII character codes:

Character	ASCII code hex	ASCII code decimal
<DLE>	10	16
\r	0d	13
\n	0a	10
>	3e	62
?	3f	63
<space>	20	32

4.3 Samples

NOTE:

- The underlined characters are only returned if echo mode is on.
- Sequences separated by | in {} mean alternatives of which exactly one is true.
- Sequences in [] can but don't have to appear.
- Italic words in <> are symbolic parameter names. Their meaning and allowed values will be explained in the following text.

Ask device identification:

<DLE>**ID**\r
>ID_<*InstrName*>;_<*Version*>;_<*SNum*>\r\n

The device answers its device type <InstrName>, the firmware-version <Version> and the serial number <SNum>. E.g. `Equotip 550;2.1.1;UP01-000-0509`

Enable, disable or request echo mode:

<DLE>**EC?**
>EC_{0|1}\r\n
<DLE>**EC**{0|1|ON|OFF}\r
>EC{0|1}_[?]\r\n

The first sample asks whether or not echo mode is on.
The second sample sets the echo mode off (`EC0` or `EC OFF`) or on (`EC1` or `EC ON`).

5. Remote ASCII commands

5.1 General remarks and rules

5.1.1 Affected measurement series

Many commands request or change the settings of a measurement (series).
All of them do only change the current series (and future series) in memory. This means if you have opened an old series with the explorer, this old series will not be changed persistently.
Example:
Your current measurement series name is `532`. Then you open an old series with the name `123` and see the series `123` on the display of the platform. With the command `SNAM` you will receive `123` and can change it to e.g. `TEST`. When you leave the explorer view and go back to measurement view, the series name will be `532` again. When you next time open `123`, the name will still be `123` and not `TEST`.

5.1.2 Active Probe

Some commands need an active probe (impact device) to be valid. To have an active probe, make sure you are in the measurement screen, the appropriate probe is connected and activated (so you can measure with that probe). If you have more than one probe connected, you can see the active probe with `PIDL`.

The following commands need an active probe:
`PAPV, PHWR, PNAM, PSNO`

In case you change settings, they will be stored context related. Therefore it makes sense to have an active probe of the appropriate type (Portable Rockwell, Equotip Leeb, UCI…) plus the correct conversion scale and material selected when you change settings (e.g. `UPPR, LOWR`…)

5.1.3 Missing commands, missing functionality

This documentation, the whole Software Package (PqRemote_V_x_y_z.zip) and the platform device firmware are subject to change. Most likely changes will be bug fixes, adding more remote commands and functionality. Therefore it's good to have the latest version:

- This document is in the document section of the device (as part of the multimedia package) plus in the Software Package.
- The Software Package can be downloaded from the Proceq homepage (www.proceq.com) or from the Link Tool provided.
- The firmware update can be started from the Link Tool.

5.2 Overview

Key for the tables below:

RL:	Is a successful remote login (`RLGI`) mandatory to execute this command?
SU:	Does the active profile in the device have to be super user? Note: This is always true for platform applications by now.
LK:	Must the device be locked (`RLCK`) to execute this command?
note:	An X in the table means this is required.

5.2.1 Alphabetical list of Platform common commands

These commands should be the same on all Platform applications (Equotip 550, Pundit PL-2, Profometer PM-6…)

Command	Short description	RL	SU	LK
EC	Switch the echo mode			
ID	Get the instrument ID (short form)			
@ID@	Get the extended Platform specific instrument ID			
DELF	Delete a file or directory	X	X	X
DIRS	Get the directory structure including file names			
DMSG	Disable a specific message type			
EMSG	Enable a specific message type			
GETF	Load a file to the PC			
GPOW	Get power info (voltage, current, battery status)			
HELP	Displays help regarding these commands			
IAPV	Get the instrument application version			
IHWR	Get the instrument hardware revision			
INAM	Get the instrument name			
IOSV	Get the instrument Operating System version			
ISNO	Get the instrument serial number			
KEYB	Lock/unlock the keyboard or simulate a key press			
MOVF	Move or rename a file or directory	X	X	X
RLCK	Lock or unlock the device	X	X	
RLGI	Remote login			
RLGO	Remote logout			
SCSH	Get a screen shot from the device display			
TIME	Get/set the current date and time			
TOUC	Send touch events to the device			

5.2.2 Alphabetical list of Equotip specific commands

These commands are for Equotip devices.

Command	Short description	RL	SU	LK
IMPD	Get/set the impact direction			
LOWR	Get/set the lower limit			
MATR	Get/set the material group			
PAPV	Get the probe application version			
PHWR	Get the probe hardware revision			
PIDS	Get the ID of all connected probes (short form)			
PIDL	Get the ID of all connected probes (long form), shows active p.			
PNAM	Get the probe name (e.g. "D")			
PSNO	Get the probe serial number			
SCAL	Get/set the conversion scale			
SDEV	Get the measurement series standard deviation			
SERM	Save and close the current series of measurements			
SLEN	Get/set the measurement series length.			
SMAX	Get the maximum value of the measurement series.			
SMEA	Get the mean value of the measurement series.			
SMIN	Get the minimum value of the measurement series.			
SNAM	Get/set the measurement series name			
SNOR	Get the number of readings in the meas. series.			
SRAN	Get the range (max – min) in the measurement series.			
SVAL	Get a reading of the measurement series.			
UPPR	Get/set the upper limit			

5.2.3 By function

Measurement

```
DMSG EMSG IMPD LOWR MATR SCAL SDEV SERM SLEN SMAX SMEA SMIN SNAM
     SNOR SRAN SVAL UPPR
```

Version

```
ID   @ID@ IAPV IHWR INAM IOSV ISNO PAPV PHWR PIDS PIDL PNAM PSNO
```

File

```
DELF DIRS GETF MOVF
```

Communication

```
EC   HELP
```

System

```
GPOW KEYB RLCK RLGI RLGO SCSH TIME TOUC
```

5.3 Detailed command description

In this subchapter, all commands are described in detail. They are listed in alphabetical order.

In the **first line** of each command description on the left you see the command, and then a short description followed by the flags RL, SU, and LK:

<CMD>	<Description>	RL:	SU:	LK:

RL: Is a successful remote login (RLGI) mandatory to execute this command?
SU: Does the active profile in the device have to be super user?
LK: Must the device be locked (RLCK) to execute this command?
NOTE: An X means this is required.

In the **second table row**, the syntax is described. If you can use the command in several ways (e.g. get, set), each way is listed separately (1. 2. 3.).
The string following Input: is what you have to use as second parameter (cmd) in PQ_SendCMD().
The string following Output: describes what the device will answer in the third parameter (reply) of PQ_SendCMD() in case echo mode (see EC) is off.
If none is following Output:, no reply from the device is expected. In this case, PQ_SendCMD() returns 0 in case of success, -1 or 1 and '?' in case of failure.

NOTE:

- The low level sequences (see 4. Remote commands low level Protocol description) are handled in PQ_SendCMD() and are not listed here.
- Words in <> are symbolic parameter names. Their meaning and allowed values will be explained in the following text, mostly in the form of where <param> = …
- ASCII string parameter like <fname> of DELF or <Name> of SNAM can and should be enclosed with quotation marks ". If you don't, characters are converted to uppercase and the parser cannot handle some special characters appropriate.

In the **third table row**, the command is described. Special cases, samples and limitations are mentioned here.

EC	Switch echo mode	RL:	SU:	LK:

Syntax

```
1. To switch echo mode on:
   Input: EC 1 or EC ON
   Output: none

2. To switch echo mode off:
   Input: EC 0 or EC OFF
   Output: none

3. To get the current echo mode:
   Input: EC ?
   Output: 0 or 1
```

Description

To turn echo mode on or off.
If echo mode is on, the device echoes the input before it's answer:

```
Input: EC ?
Output: EC ? 1
```

NOTE:

- Echo mode is off by default.
- If echo mode was off and is set on, the echo is turned on after the command (the device answer is none)
- If echo mode was on and is set to off, the echo is turned off after the command (the device answers `EC 0`)

ID	Get instrument ID	RL:	SU:	LK:

Syntax

```
Input: ID
Output: <instrument ID>  (e.g. Equotip 550;2.1.1;UP01-000-0509)
where   <instrument ID>  = instrument name; application version; serial number
```

Description

Gets basic information about the instrument.
This command is supported by all newer Proceq devices.
Opposing to `@ID@`, the information fields (instrument name …) are the same for each Proceq device.

@ID@	Get extended instrument ID	RL:	SU:	LK:

Syntax

```
Input: @ID@
Output: <extended instrument ID>
where   <extended instrument ID>  =
        - instrument name;
        - hardware revision;
        - serial number;
        - signature;
        - application version;
        - OS version;
        - Bootloader version;
        - Multimedia package version;
        - current localization; (TLA)
        - timezone ID (4 Byte integer as ASCII hex with 0x preamble)

Sample output:
   Equotip 550;C1;UP01-000-0509;0D020200;2.1.1;2.1.0;1.2.0;1.0.0;ENU;0x0000053c
```

Description

Gets extended information about the instrument.
This command returns device specific information fields. The fields listed here are true for Platform devices.

DELF	Delete file	RL:X	SU:X	LK:X

Syntax

```
Input:  DELF "<fname>"
where   <fname> = file or directory name with full path
Output: none
```

Description

To delete a file on the device you are connected to.

NOTE:

- You must be logged in and super user and the device must be locked.
- Use this command with caution, the file gets deleted immediately and cannot be restored!
- Use `\\` instead of `\` e.g. "`\\0D020200\\0D020200.log`"
- Use the `DIRS` command to get the real file names. The file names shown on device Explorer could be only a part of the real file names.
- To access the measurement series root "`\`" on Equotip 550, use the folder "`\\0D020200\\measurements\\`"
- You can only delete empty directories.

DIRS	Display directory structure	RL:	SU:	LK:

Syntax

```
1. To get all file types:
   Input:  DIRS
   Output: <len>;<flags>;<level>;"<name>";<flags>;<level>;"<name>";...
   where   <len>   = total answer length (8 char hex)
           <flags> = xyzz: x(1 is directory), y(file flags), zz(file type)
           <level> = 00=root, 01=1 below root, 02= 2 below root...
           <name>  = file or directory name
2. To get specific file types:
   Input:  DIRS <file type>
   where   <file type>  = *NOT SUPPORTED BY NOW*, returns 0x0000
                          any combination of file types you want to get:
                 0x0001 = Measurement series
                 0x0002 = Conversions
                 0x0004 = Language DLLs
                 0x0008 = Help files
                 0x0010 = The user profile
                 0x0020 = Log files
   Output: <len>;<flags>;<level>;"<name>";<flags>;<level>;"<name>";...
   where   All fields as described under 1.
```

Description

To get information about the files stored in the device. You can either get all files and directories or a combination of specific file types.
E.g. if you want to get information about the conversion and the language DLL files you send `DIRS 0x0006`. FILE TYPES NOT SUPPORTED BY NOW

To save space, the absolute path to each file is omitted. Instead the file and folder structure is transmitted recursive and the location of a specific file is defined by its level and preceding directories.
It's best to do a sample. Find below the folder and file structure of a sample Equotip 550 device:

`\0D020200\measurements\`	Root folder for measurements
A	Subdirectory `A`
AA	Subdirectory `AA` below directory `A`
7	File `7` in subdirectory `AA`
1	File `1` in subdirectory `A`
8	File `8` in subdirectory `A`
B	Subdirectory `B`
4	File `4` in subdirectory `B`
6	File `6` in subdirectory `B`
2	File `2` in measurement root directory
3	File `3` in measurement root directory
5	File `5` in measurement root directory

If you send `DIRS 0x0001` to the above device, the answer is: `00000000;` FILE TYPES NOT SUPPORTED BY NOW. If you send `DIRS`, you will get something like this:

```
000001FC;1000;00;"\";1000;01;"0D020200";1000;02;"measurements";0000;03;"2#_0_1
_2_7893_1_0.eq5";0000;03;"3#_0_1_2_7758_1_0.eq5";0000;03;"5#_0_1_2_7685_1_0.eq
5";1000;03;"A";1000;04;"AA";0000;05;"7#_0_1_2_7781_1_0.eq5";0000;04;"8#_0_1_2_
8521_1_0.eq5";0000;04;"1#_0_1_2_7879_1_0.eq5";1000;03;"B";0000;04;"6#_0_1_2_76
84_1_0.eq5";0000;04;"4#_0_1_2_8427_1_0.eq5";1000;02;"calib";0000;03;"Verif-
ID51-006-0169#_21_1_2_7628_10_1.eq5";0000;03;"Verif-ID51-006-
0169#_39_1_2_7620_10_1.eq5";1000;02;"curves";0000;03;"Standard5.cnv";
```

DMSG	Disable message	RL:	SU:	LK:

Syntax

```
1. To disable a specific message <msg>:
   Input:  DMSG <msg>
   where   <msg> = message type number to disable (1...4)
   Output: none

2. To check if <msg> is disabled:
   Input:  DMSG <msg>?
   Output: Y = message is disabled, N = message is enabled.

In Equotip Application version 2.8.0 and newer (above still supported):

3. To disable all supported message types:
   Input: DMSG 9999 Y
   Output: none
```

Description

Disables a specific message type (or all messages) previously enabled by EMSG.
See EMSG for further information.

EMSG	Enable message	RL:	SU:	LK:

Syntax

```
1. To enable a specific message <msg>:
   Input:  EMSG <msg>
   where  <msg> = 1: Limit over-/underrun message
          <msg> = 2: Probe removed/connected message
          <msg> = 3: Begin of measurement procedure message
          <msg> = 4: New measurement (value included) message
   Output: none

2. To check if <msg> is enabled:
   Input:  EMSG <msg>?
   Output: Y = message is enabled, N = message is disabled.

In Equotip Application version 2.8.0 and newer (above msg still supported):

1. To enable a specific message <msg>:
   Input:  EMSG <msg>
   where  <msg> = 0: Error message (value key see separate table)
   where  <msg> = 8: The status during a measurement (key see separate table)
   where  <msg> = 9: Intermediate values during the measurement
   where  <msg> =10: New measurement secondary scale (only if enabled)
   Output: none

3. To enable all supported message types:
   Input: EMSG 9999 Y
   Output: none
```

SAMPLES:

EMSG 1 = Gives an output if the limits are over- resp. underrun:

<A:HI

<A:LO

EMSG 2 = Probe: Outputs removed/connected probe:

<P:

<P:Equotip Leeb Impact Device D

EMSG 3 = Gives an output at the beginning of a measurement, even when there was an error:

<C

EMSG 4 = Outputs new valid values including Scale and limit over- underruns:

<M:709 HLD +

Even though errors like "Signal not evaluable" are not implemented with App version 2.7.0 and earlier, one could find out measurement related errors by enabling (1,) 3, 4:

Good measurement:

<C

<M:630 HLD

Above upper limit:

<C

<M:709 HLD +

<A:HI

Error:

<C

SAMPLES from App Version V2.8.0 and beyond:

Good portable Rockwell measurement with status (<S:) and secondary scale (<M2:) enabled:
<C
<S:4
<S:5
<S:6
<M:9.8 µm
<M2:789 HLD

Bad portable Rockwell measurement with error (<E:) incomplete
<C
<S:4
<E:2

Good UCI measurement with status (<S:), intermediate values (<I:) and secondary scale (<M2:) enabled:
<C
<S:1
<I:UCI, 0 df, 0 N/100
<I:UCI, 0 df, 0 N/100
<I:UCI, 219 df, 1273 N/100
<I:UCI, 332 df, 1920 N/100
<I:UCI, 522 df, 3169 N/100
<I:UCI, 728 df, 4953 N/100
<S:2
<I:UCI, 837 df, 6084 N/100
<I:UCI, 959 df, 7036 N/100
<I:UCI, 868 df, 3765 N/100
<I:UCI, 394 df, 0 N/100
<I:UCI, 0 df, 0 N/100
<I:UCI, 0 df, 0 N/100
<I:UCI, 0 df, 0 N/100
<I:UCI, 0 df, 0 N/100
<M:807 HLD
<M2:708 HV

Format of <I:<probe type>, <shift frequency>, <force in N/100> where only supported probe type is UCI.

Bad UCI measurement with status (<S:), intermediate values (<I:) and error (<E:) incomplete
<C
<S:1
<I:UCI, 0 df, 0 N/100
<I:UCI, 9 df, 0 N/100
<I:UCI, 632 df, 4057 N/100
<E:2

Error message codes (message type 0)

For almost all errors, the measurement is aborted and no <M: and <M2: will be issued.
L = supported for Leeb, R = supported for portable Rockwell, U = supported for UCI

#	*Short description*	*L*	*R*	*U*	*Comment*
1	Bad measurement	X		X	The only error issued for Leeb measurements, but is not always issued on way off (air) Leeb measurements (check <C too)
2	Incomplete		X	X	The measurement did not pass/go successful through all steps
3	Thin			X	The material is considered too thin, measurement not possible.
4	Overload			X	The applied force is too high
5	Idle frequency			X	→Bad measurement
6	Premature			X	→Bad measurement
7	Communication			X	Communication error with UCI probe
8	Material			X	The material is considered too soft
9	Init			X	A reinitialization of the probe seem to be necessary. This only occurs if the past measurements all failed. Note: once issued, this requires user action on device!!! Must not happen in automation or you can try to answer using TOUC.
10	Excessive load			X	Too much force was applied, apply less load or the probe might be seriously damaged.
11	Too less samples		X		
12	Too fast down		X		
13	Too fast released		X		
14	Vibration warning		X		This is only a warning, so the measurement is considered valid.

Status message codes (message type 8)

R = supported for portable Rockwell, U = supported for UCI (Leeb does not have any monitored states)

#	*Short description*	*R*	*U*	*Comment*
0	Idle	X		Before/at start of a measurement, is not sent remotely
1	Touches		X	The probe touches the material under test
2	Force applied		X	The required force is applied
3	Init done		X	The probe is reinitialized after error init (see there)
4	Hold	X		Hold the probe with current force in current position
5	Release	X		Release the probe from the material under test
6	Finished	X		The measurement is finished

Description

To enable / activate asynchronous messages of a specific type.
Once enabled, the device sends information over the interface for which the message was enabled, as soon as the event (e.g. new measurement) occurred.

To get that information, you have to check for incoming data with PQ_Read() or a lower layer function in case you don't use Pq_Remote in your application.

IMPORTANT NOTE:

Because the measurement, UI (and other) tasks in the device have higher priority than the communication task, **we cannot guarantee a maximum latency time!** This means a status message to e.g. release the force might be received with a delay of seconds. Make sure you take this into account within your project.

GETF	Load a file to the PC	RL:	SU:	LK:

Syntax

```
Input: GETF <fname>
where  <fname>  = the full filename inclusive path

Output: <len> [end ASCII reply] <blen><file>
where   <len>   = file length in ASCII hex
        <blen>  = file length (4 byte binary)
        <file>  = the file (binary data stream)
```

Description

To load a file, usually a measurement series, from the device to the PC.

You will receive <len> in the reply of PQ_SendCMD() and <blen> and <file> in the following call to PQ_Read()

NOTE

- Use \\ instead of \ e.g. "\\0D020200\\0D020200.log"
- Enclose <fname> with quotation marks ".
- Use DIRS to get the list of available files.

GPOW	Get power info	RL:	SU:	LK:

Syntax

```
Input: GPOW <param>
where  <param> = 0: allow cached values
       <param> = 1: force refresh of values
       <param> =  : default behaviour (0)
Output: <bV>;<bI>;<bT>;<blp;<bF>;<bC>;<aS>
where   <bV>  = battery Voltage in ASCII dec       eg. 3765 mV
        <bI>  = battery Current in ASCII dec       eg. 1370 mA
        <bT>  = battery Temperature in ASCII dec   eg. 40.2 °C
        <blp> = battery life percent in ASCII dec  eg. 68
        <bF> = battery flags BATTERY_FLAG_UNKNOWN,BATTERY_FLAG_HIGH,
                             BATTERY_FLAG_LOW,BATTERY_FLAG_CRITICAL,
                             BATTERY_FLAG_CHARGING
        <bC> = battery chemistry BATTERY_CHEMISTRY_LION, WRONG_VALUE
        <aS>  = AC line status  AC_LINE_UNKNOWN,AC_LINE_OFFLINE,AC_LINE_ONLINE
```

Description

To get information about the power status of the device.

NOTE

- In case the AC adapter is connected (AC_LINE_ONLINE), battery flags and battery life percent cannot be evaluated

HELP	Displays help on ASCII commands	RL:	SU:	LK:

Syntax

```
1. To get a command overview:
   Input:  HELP
   Output: none to PQ_SendCMD(), the available Commands to PQ_Read()

2. To get detailed help on a specific command:
   Input:  HELP <cmd>    (e.g. HELP IMPD)
   Output: none to PQ_SendCMD(),
           the detailed help for <cmd> to PQ_Read()
```

Description

To get help on the available ASCII commands.
This is especially helpful if you play around with PqRemoteDemoxxx.exe

NOTE

- We can not send the output to PQ_SendCMD(), because the answer from the device contains several \r\n escape characters. Each of those stops parsing the answer. You will receive the answer with PQ_Read().

IAPV	Get the application version	RL:	SU:	LK:

Syntax

```
Input:  IAPV
Output: instrument application version (e.g. 1.5.0)
```

Description

To get the version of the currently running application.
You can upgrade / check the latest Application with PqUpgrade (out of the Link tool).

IHWR	Get the hardware revision of the instrument	RL:	SU:	LK:

Syntax

```
Input:  IHWR
Output: instrument hardware revision (e.g. A3)
```

Description

To get the hardware revision of your indicating device.

IMPD	Get/set impact direction	RL:	SU:	LK:

Syntax

```
1. To set impact direction:
   Input: IMPD <Direction>
   where <Direction> = A:   automatic determination of direction
         <Direction> = 0:   vertical down
         <Direction> = 45:  diagonal down
         <Direction> = 90:  horizontal
         <Direction> = 135: diagonal up
         <Direction> = 180: vertical up
   Output: none

2. To get impact direction:
   Input: IMPD ?
   Output: <Direction> (values see above)
```

Description

If you set a new impact direction, it will affect future measurements, but it will not affect the last measurement (direction is stored for each measurement).

INAM	Get the instrument name (type)	RL:	SU:	LK:

Syntax

```
Input:  INAM
Output: instrument name (e.g. Equotip 550)
```

Description

To get the name / type of your indicating device.

IOSV	Get the instrument OS version	RL:	SU:	LK:

Syntax

```
Input:  IOSV
Output: instrument OS version (e.g. 1.1.5)
```

Description

To get the operating system version of your indicating device.

ISNO	Get the instrument serial number	RL:	SU:	LK:

Syntax

```
Input:  ISNO
Output: instrument serial number
          e.g. UP01-000-0509-14/3P
          where 14 = year of production, 3P = quarter of production +
producer)
```

Description

To get the serial number of your indicating device.

KEYB	Keyboard I/O	RL:	SU:	LK:

Syntax

```
1. To lock resp. unlock the instrument keyboard:
   Input: KEYB LO[CK] resp. KEYB UN[LOCK]
   Output: none
2. To simulate a key press with fixed short pressed duration:
   Input: KEYB <Key>
   where <Key> = OFF:   simulate pressing the ON/OFF/HOME key
         <Key> = DISP:  simulate pressing the DISPLAY key
         <Key> = BACK:  simulate pressing the BACK key
   Output: none
3. To simulate a key press with remote controlled pressed duration:
   Input: KEYB A:PRESS;<Key>   press down <Key>
   Input: KEYB A:RELEASE;>Key> release <Key>
   Output: none
```

Description

With this command you can simulate the device keyboard and therefore remotely control the device e.g. together with `SCSH`.
This command is used in the sample DemoRemote.exe

NOTE:

- If you locally lock the keyboard with `KEYB LO`, keypresses sent with `KEYB` will still be processed.
- If you use the remote controlled pressed duration, make sure you don't forget to release the key!

LOWR	Get/set lower limit	RL:	SU:	LK:

Syntax

```
1. To set lower limit:
   Input: LOWR <Limit>
   where <Limit> = OFF:     no limitation
         <Limit> = xxx[.x]: limit value
   Output: none

2. To get lower limit:
   Input: LOWR ?
   Output: <Limit> (values see above)
           <Limit> = noCnv: conversion not possible
```

Description

If you set the lower limit, it will affect all measurements in the current series (and future series), also the ones already measured.

MATR	Get/set material	RL:	SU:	LK:

Syntax

```
1. To set material:
   Input: MATR <Mat>
   where <Mat> = 1,2, ..., 12: standard material group
         <Mat> = C:"<MatName>"  customer defined conversion
   Output: none (=ok) or ? in case of error

2. To get material:
   Input: MATR ?
   Output: <Mat> (values see above)
```

Description

This will behave as if you change the material locally. Only materials will be accepted which are valid for the current probe and scale. If you try to set an invalid material, you'll get an error (?).

Make sure you are in the measurement mode/screen with an activated probe and do not have the Measurement Settings open when using MATR. Else this command will not work properly.

MOVF	Move/rename a file or directory	RL:X	SU:X	LK:X

Syntax

```
Input: MOVF "<fnam>";"<new fnam>"
where <fnam>      = the existing file or directory + full path
      <new fnam>  = the new location and or name
Output: none

Examples:
movf "\\0D020200\\measurements\\A\\AA\\";
     "\\0D020200\\measurements\\A\\AB\\"    → this renames AA in AB

movf "\\0D020200\\measurements\\A\\AB\\";
     "\\0D020200\\measurements\\AB\\"    → moves AB plus content one up
```

Description

You can also move/rename a directory that is not empty.

NOTE:

- Make sure you are logged in, super user and the device is locked, else you receive?

PAPV	Get the probe application version	RL:	SU:	LK:

Syntax

```
Input:  PAPV
Output: probe application version (e.g. 0.3.1)
```

Description

To get the hardware revision of the connected and activated probe. Will return an empty string if no probe is connected and activated or the probe has no firmware.

PHWR	Get the probe hardware revision	RL:	SU:	LK:

Syntax

```
Input:  PHWR
Output: probe hardware revision (e.g. A0)
```

Description

To get the hardware revision of the connected and activated probe. Will return an empty string if no probe is connected and activated.

PIDS	Get short ID of all connected probes	RL:	SU:	LK:

Syntax

```
Input: PIDS
Output: [< probe ID>][;<probe ID>]…
where   < probe ID>  = probe name; firmware version; serial number

Sample output:
   Equotip Leeb Impact Device D;;ID51-006-0169;Portable Rockwell
   Probe;0.3.1;ES01-000-0009
```

Description

Gets basic information about all the connected probes.

PIDL	Get long ID of all connected probes	RL:	SU:	LK:

Syntax

```
Input: PIDL
Output: [<extended probe ID>][;<extended probe ID>]…
where   <extended probe ID>  =
        - probe name;   [active]= this is the active probe in use
        - hardware revision;
        - serial number;
        - signature;
        - firmware version;

Sample output:
   Equotip Leeb Impact Device D [active];A;ID51-006-0169;81-00000-0;;Portable
   Rockwell Probe;A2;ES01-000-0009;08000000;0.3.1
```

Description

Gets extended information about all the connected probes.
This command returns device application specific information fields. The fields listed here are true for Platform Equotip probes.

PNAM	Get the probe name (type)	RL:	SU:	LK:

Syntax

```
Input:  PNAM
Output: probe name (e.g. D)
```

Description

To get the name / type of your connected and activated probe. Will return an empty string if no probe is connected and activated.

PSNO	Get the probe serial number	RL:	SU:	LK:

Syntax

```
Input:  PSNO
Output: Probe serial number
          e.g. ID51-006-0169
```

Description

To get the serial number of the connected and activated probe. Will return an empty string if no probe is connected and activated.

RLCK	Lock / unlock the device	RL:X	SU:X	LK:

Syntax

```
1. To lock the device:
   Input: RLCK ON
   Output: ON

2. To unlock the device:
   Input: RLCK OFF
   Output: OFF

3. To get the current lock state:
   Input: RLCK ?
   Output: ON or OFF
```

Description

To lock the device. This will pop up a dialog and any local input will be blocked. This command is only needed for few critical remote commands like `DELF` or `MOVF`

NOTE:

- Must be logged in and super user.

RLGI	Remote login	RL:	SU:	LK:

Syntax

```
Input:  RLGI [<UserName>] [;<Password>] [;<Timeout>]
Output: <User Name>;<Flags>
where  <Flags>:   0001 = The active user has a password
                  0002 = The active user is a super user (administrator)
                  0004 = The login was successful
       <Timeout>: automatically logout after <Timeout> in[ms]
```

Description

To remotely log into the device. Is needed for few critical remote commands like `DELF` or `MOVF`

NOTE:

- For all platform devices UserName and Password can be empty. E.g. `RLGI` is enough.
- Any login attempt to a not active user is rejected.
- Multiple `<Flags>` can be combined. E.g 0006 means super user and logged in.
- If the active user has no password defined, sending `RLGI` without parameters will be enough to log in.
- If you specify a timeout, `RLGI` must be sent before timeout to reset the automatic logout timer. If you don't specify a timeout, you will remain logged in as long as the device doesn't shut down or you didn't send `RLGO`

RLGO	Remote logout	RL:	SU:	LK:

Syntax

```
Input:  RLGO
Output: none
```

Description

To remotely log out.

SCAL	Get or set the conversion scale	RL:	SU:	LK:

Syntax

```
1. To set scale:
   Input: SCAL <Scale>
   where <Scale> = "HB", "HV", "HRC", "um", "in/1000"...
   Output: none (ok) or ? in case of invalid scale

2. To get scale:
   Input: SCAL ?
   Output: <Scale> (values see above)
```

Description

This will behave as if you change the scale locally. Only scales will be accepted which are valid for the current probe and material. If you try to set an invalid scale, you'll get an error (?).

In case you use a portable Rockwell probe and use metric system, you can send SCAL "in/1000" and it will switch to imperial and native scale. The same is true if you use imperial and send SCAL "um", it will switch to metric and native scale.

Make sure you are in the measurement mode/screen with an activated probe and do not have the Measurement Settings open when using MATR. Else this command will not work properly.

NOTE: The scale is case sensitive, so use ". E.g. SCAL um will receive ?, SCAL "um" will work.

SCSH	Screen shot	RL:	SU:	LK:

Syntax

```
1. To get the screen shot:
   Input: SCSH
   Output: <param>;<len> [end ASCII reply] <bparam><blen><bimage>
   where   <param>  = ASCII hex bitfield to set image parameters:
                      0000 = BMP bitmap
                      0001 = GIF bitmap
           <len>    = the length in ASCII hex of the following bitmap
           <bparam> = the param, see above (2 byte binary)
           <blen>   = the length of the bitmap (4 byte binary)
           <bimage> = the raw image data of the screenshot
                      (binary byte stream)

2. To get the current param settings
   Input: SCSH ?
   Output: <param>  = ASCII hex bitfield with current image parameters

3. To set the current param settings
   Input: SCSH <param>
   where   <param>  = ASCII hex bitfield with the new image parameters
   Output: none
```

Description

To transfer a screen shot to the PC. The ASCII output will be received in PQ_SendCMD(), the binary part must be read out afterwards with PQ_Read().
This command is used in the sample DemoRemote.exe

SDEV	Measurement series standard deviation	RL:	SU:	LK:

Syntax

```
Input:  SDEV ?
Output: Series standard deviation or ? or noCnv when conversion not possible
```

Description

To delete a (specific) measurement.

NOTE:

- `SDEV` will be rejected ('?') when no readings are available in the series.
- If only one measurement is available, `SDEV` will answer with ('-')
- Series can be open or closed.

SERM	Close and save current series of measurements	RL:	SU:	LK:

Syntax

```
Input: SERM CLOSE
Output: none or ? in case of error
```

Description

Saves and closes the current measurement series and starts a new one.

NOTE:

- Only works if you are in measurement mode and the current series has at least one measurement.

SLEN	Get/set series length	RL:	SU:	LK:

Syntax

```
1. To set an unlimited series length (i.e. series have to be closed manually):
   Input: SLEN -
   Output: none

2. To set a specific series length:
   Input: SLEN <Length>
   where <Length> = 1 .. 9999
   Output: none

3. To get the series length:
   Input: SLEN ?
   Output: <Limit> (- or 1 .. 9999)
```

Description

The series length is the number of measurements until the series is closed automatically.

NOTE:

- When you set SLEN to a specific length, auto close series action will be set to save and next.
- When you set SLEN to unlimited, auto close series will be set to save. See SERM for remote save and close a series with unlimited length.
- Closing and saving the series takes some time. Delay the next measurement accordingly.
- If you change SLEN, it will not affect an already closed series.
- If you change SLEN to a value smaller than the actual measurement number of an open series, the actual series will not be closed automatically.
- See SNOR, too.

SMAX	Get the series maximum value	RL:	SU:	LK:

Syntax

```
Input:  SMAX ?
Output: Series maximum value or --- or noCnv or ? when conversion is not
        possible
```

Description

NOTE:

- SMAX will be rejected when no readings are available in the series.
- Series can be open or closed.

SMEA	Get the series mean	RL:	SU:	LK:

Syntax

```
Input:  SMEA ?
Output: Series mean value or --- or noCnv or ? when conversion not possible
```

Description

NOTE:

- SMEA will be rejected when no readings are available in the series.
- Series can be open or closed.

SMIN	Get the series minimum value	RL:	SU:	LK:

Syntax

```
Input:  SMIN ?
Output: Series minimum value or --- or noCnv or ? when conversion is not
        possible
```

Description

NOTE:

- SMIN will be rejected when no readings are available in the series.
- Series can be open or closed.

SNAM	Get/set series name	RL:	SU:	LK:

Syntax

```
1. To set the series name:
   Input: SNAM "<Name>"
   Output: none

2. To get the series name:
   Input: SNAM ?
   Output: <Name>
```

Description

NOTE:

- When a series is already saved, it will not be resaved after setting a new name.
- Some characters are not allowed: "/:*<\>|

SNOR	Get series number of readings	RL:	SU:	LK:

Syntax

```
Input:  SNOR ?
Output: Series number of readings
```

Description

NOTE:

- noCnv measurements are also counted.
- See SLEN, too.

SRAN	Get series range	RL:	SU:	LK:

Syntax

```
Input:  SRAN ?
Output: Series range or noCnv when conversion not possible
```

Description

NOTE:

- SRAN will be rejected when no readings are available in the series.

SVAL	Get a reading / measurement	RL:	SU:	LK:

Syntax

```
1. To get the latest reading of the current series:
   Input: SVAL ?
   Output: value of the reading

2. To get a specific reading of the current series:
   Input: SVAL <index>
          where <index> = 1 .. number of readings
   Output: value of the reading or noCnv when conversion not possible
```

Description

NOTE:

- SVAL will be rejected when no readings are available in the series.

TIME	Get/set date and time	RL:	SU:	LK:

Syntax

```
1. To get the current date/time of the instrument:
   Input: TIME
   Output: date and time in the format <dd>-<mmm>-<yyyy>; <hh>:<mm>:<ss>
   e.g. 07-JUL-2008; 08:50:22

2. To set the current date/time of the instrument:
   Input: TIME  <dd>-<mmm>-<yyyy>; <hh>:<mm>:<ss>
   Output: none
```

Description

It is the local time you get/set, not UTC, GMT.

TOUC	Simulate device touch events	RL:	SU:	LK:

Syntax

```
Input: TOUC <o>;<n>;<x0>;<y0>;…;<xi>;<yi>
Output: none or ? in case of error
   where   <o>  = number of fingers pressed before (old)
           <n>  = number of fingers pressed now (new)
           <x0> = x coordinate of first finger (0..65535)
           <y0> = y coordinate of first finger (0..65535)
           <xi> = x coordinate of ith finger (0..65535), i = max(o,n)
           <yi> = y coordinate of ith finger (0..65535), i = max(o,n)
```

Description

This command sends touch events to the device.

Samples:

`TOUC 0;1;<x0>;<y0>` = One finger is pressed

`TOUC 1;1;<x0>;<y0>` = One finger is still pressed and has moved

`TOUC 1;0;<x0>;<y0>` = One finger has left the screen (been released)

`TOUC 1;2;<x0>;<y0>;<x1>;<y1>` = One finger was and a 2nd is newly pressed

NOTE:

- The top left of the screen maps to coordinates (0,0) the bottom right maps to (65535,65535).
- When fingers are released (e.g. `TOUC 1;0;<x0>;<y0>` or `TOUC 2;0;<x0>;<y0>;<x1>;<y1>` or `TOUC 2;1;<x0>;<y0>;<x1>;<y1>`), you send the last known, old coordinates of the finger(s). In all other cases, you send the actual coordinates.

UPPR	Get/set upper limit	RL:	SU:	LK:

Syntax

```
1. To set upper limit:
   Input: UPPR <Limit>
   where <Limit> = OFF:     no limitation
         <Limit> = xxx[.x]: limit value
   Output: none

2. To get upper limit:
   Input: UPPR ?
   Output: <Limit> (values see above)
           <Limit> noCnv:   conversion not possible
```

Description

If you set the upper limit, it will affect all measurements in the current series (and future series), also the ones already measured (it's stored once per series).

Proceq Europa
Ringstrasse 2
CH-8603 Schwerzenbach
Phone +41-43-355 38 00
Fax +41-43-355 38 12
info-europe@proceq.com

Proceq UK Ltd.
Bedford i-lab, Priory Business Park
Stannard Way
Bedford MK44 3RZ
United Kingdom
Phone +44-12-3483-4515
info-uk@proceq.com

Proceq USA, Inc.
117 Corporation Drive
Aliquippa, PA 15001
Phone +1-724-512-0330
Fax +1-724-512-0331
info-usa@proceq.com

Proceq Asia Pte Ltd
12 New Industrial Road
#02-02A Morningstar Centre
Singapore 536202
Phone +65-6382-3966
Fax +65-6382-3307
info-asia@proceq.com

Proceq Rus LLC
Ul. Optikov 4
korp. 2, lit. A, Office 410
197374 St. Petersburg
Russia
Phone/Fax + 7 812 448 35 00
info-russia@proceq.com

Proceq Middle East
P. O. Box 8365, SAIF Zone,
Sharjah, United Arab Emirates
Phone +971-6-557-8505
Fax +971-6-557-8606
info-middleeast@proceq.com

Proceq SAO Ltd.
Rua Paes Leme, 136, cj 610
Pinheiros, São Paulo
Brasil Cep. 05424-010
Phone +55 11 3083 38 89
info-southamerica@proceq.com

Proceq China
Unit B, 19th Floor
Five Continent International Mansion, No. 807
Zhao Jia Bang Road
Shanghai 200032
Phone +86 21-63177479
Fax +86 21 63175015
info-china@proceq.com

www.proceq.com

820 356 07 ver 2021