SM15K - Programming Manual for Ethernet & Sequencer

- **General information**
- **Installation manual**
- **Command description**

Firmware version P0108/P0109

Table of Contents:

1 General Instructions

1.1 Features

The Remote Ethernet programming function is an interface between a TCP/IP network and the Power Supply. It allows the operator to create fully automated systems. The Ethernet interface can set and measure the parameters of the power supply, read the status signals, set controls, interacts with digital I/O and generate waveforms, stored in memory. It is even possible to take over tasks from a PLC. That makes processes and test systems more powerful and less complex.

1.2 Analog

The power supplies of Delta Elektronika are very stable and accurate. The Ethernet interface is designed especially for these kinds of power supplies. Therefore the programming and measuring resolution is 16 bits for current and voltage, and 12 bits for power and all units are calibrated very precisely by the factory. To calculate the step size in Voltage or Ampere use the equation below:

$$StepSize = \frac{MaximumOutput}{2^{16}}$$

To calculate the step size in Watt use the equation below:

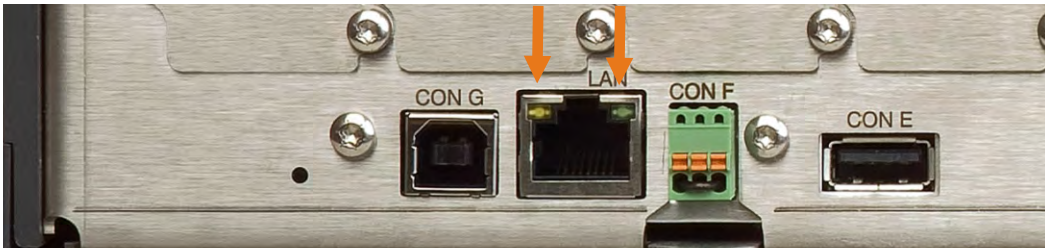
$$StepSize = \frac{MaximumOutput}{2^{12}}$$

1.3 Status

The power supplies are equipped with a lot of signals which inform the user about the status of the supply. Status like Limit, OverTemperature, DCF, etc can be read to check the condition of the system.
For example, ACF-signal (which indicates an incorrect input voltage) can be monitored, so action can be taken to avoid problems before a system is switched off.

1.4 LEDs

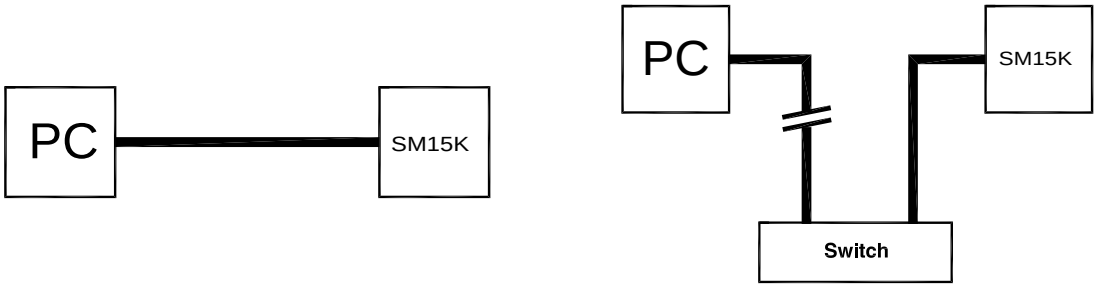
Next to the RJ45 connector are two LEDs, see picture below.
The left orange LED is named "ERR". When this LED is ON, the Ethernet interface has generated an error message as a result of a bad command or parameter.
The right Green LED is named "LNK/ACT", which stands for Link/Activity. This LED indicates whether or not the Ethernet interface is connected and communicates.



2 Installation

2.1 Infrastructure

To control or program the power supply, the Ethernet interface should be connected to a TCP/IP network, using the RJ45 connector at the rear (connector LAN). A standard RJ45 patch- or cross-link cable can be used to connect the Ethernet interface to a network switch, or directly to a PC. The Ethernet interface of the SM15K is Auto-MDIX. This means that for example a patch cable can be used for a direct connection, instead of using a cross-link cable.The "LNK/ACT"-LED of the Ethernet interface will be on when the connection is okay.



Either Cross-Link or standard Ethernet cable can be used.
The command "PING <address>" of Windows® can be used (via menu Start - RUN - CMD) to check the connection. For example : PING 10.1.0.101

2.2 Settings

Each network has it's own range of IP addresses. To be able to control the SM15K via a particular network, the IP address of the Ethernet interface must be within the address range of that network.

Recommended address ranges are:

Class A : 10.0.0.0	to	10.255.255.255	(mask : 255.0.0.0)
Class B : 172.16.0.0	to	172.31.255.255	(mask : 255.240.0.0)
Class C : 192.168.0.0	to	192.168.255.255	(mask : 255.255.0.0)

With exception of all addresses ending with '0' or '255'.
Warning : Addresses above 223.255.255.255 are not supported on several operating systems.

The TCP/IP settings of the SM15K can be set using the front panel, the internal web page[1], or using the DE Configurator program[2]. The default setting is DHCP.

1) The PC has to be set to the IP range of the power supply to view the internal web page.

2) The DE Configurator program is available for download on http://www.DeltaPowerSupplies.com/
Please see the Downloads section under, Products, SM15K.

3 Communication

3.1 Settings

There are two important settings to communicate properly with the SM15K, which are:
IP address : default DHCP. (refer to section 2.2 for settings).
Port number : fixed to 8462.

3.2 TCP/IP

Any programming language or application that can send and receive TCP/IP packages can be used for communication with the SM15K. Use socket communication on port 8462.

4 Conventions

The SM15K has a command set, which includes commands compatible with the SCPI language (Standard Commands for Programmable Instruments).

4.1 Syntax

The command descriptions contain the syntax of the instructions and show the form in which these commands can be sent to the SM15K.
It is allowed either to use the short form or the entire command.
For example : **SOURce:VOLtage<sp>5<term>** can be send as :
sour:vol<sp>5<term>
source:volt<sp>5<term>
source:voltage<sp>5<term>
sour:voltage<sp>5<term>
SoUrCe:VoLt<sp>5<term>
(Sending mixed upper- and lowercase is allowed).

4.2 Query

Commands ending with a question mark "?" (ASCII character 3FH, 63d) are interpreted as a query. If it is a valid command, the unit will respond with an answer. Otherwise an error is generated.

4.3 Space <sp>

Within the syntax spaces are used, indicated by <sp> (ASCII character 20H, 32d).

4.4 Terminator <term>

At the end of each command or query, a terminator character is required, indicated by <term>. Each reply on a valid query will end with <term>. The default terminator is a linefeed (ASCII character 0AH, 10d).
For other termination options see page 5-10.

4.5 Parameters

Within this document, parameters are used to indicate the form of data sent to or coming from the

SM15K.

<NR1>	= positive integers : 0,1,2,3,...
<NR2>	= floating point : 3.22, 0.06, etc.
<boolean>	= False or True parameter : 0 or 1, OFF or ON.
[ ]	= It is allowed to use or to skip this part of the command.

5 Command description

5.1 General instructions

*IDN?

This command is used to read the identification string of the SM15K. The string contains the name, the option number, the version of the firmware and the serial number of the Power Supply.
Syntax : ***IDN?<term>**

The response string has 5 fields, separated by commas.
For example :
DELTA<sp>ELEKTRONIKA<sp>BV,SM500-CP-90,000010207248,H0_P0102,0<term>
The first field shows the manufacturer's name.
The second field shows the power supply type.
The third field shows the serial number of the Power Supply
The fourth field shows the Firmware version of the Power Supply
The last field is reserved for future implementations.

*PUD

PUD is an abbreviation for Protected User Data. This command allows the user to give the power supply his own name for identification or to store relevant data.
For example names like "Motor Controller Setup 3", "Battery Simulator" or "Calibrated May 3rd 2017".
This information can be maximum 72 characters long and is stored into non-volatile memory (see command *SAV).
Syntax : ***PUD<sp><data><term>** data = A-Z, a-z, 0-9, <sp>, _ and -, 72 maximum

To read the Protected User Data:
Syntax : ***PUD?<term>**

*SAV

To store all relevant settings, this command saves them into non-volatile memory.

A summary of the saved setting are shown below:

Calibration gain current measure	: CALIbrate:CURrent:MEAsure:GAIn
Calibration offset current measure	: CALIbrate:CURrent:MEAsure:OFFset
Calibration gain voltage measure	: CALIbrate:VOLtage:MEAsure:GAIn
Calibration offset voltage measure	: CALIbrate:VOLtage:MEAsure:OFFset
Protected User Data	: *PUD
Password	: SYSTem:PASsword

If an INT MOD ANA (optional) is installed:

Calibration gain current programming	: CALIbrate:INTerface:GAIn IPRG
Calibration offset current programming	: CALIbrate:INTerface:OFFset IPRG
Calibration gain voltage programming	: CALIbrate:INTerface:GAIn VPRG
Calibration offset voltage programming	: CALIbrate:INTerface:OFFset VPRG
Calibration gain current monitoring	: CALIbrate:INTerface:GAIn IMON
Calibration offset current monitoring	: CALIbrate:INTerface:OFFset IMON
Calibration gain voltage monitoring	: CALIbrate:INTerface:GAIn VMON
Calibration offset voltage monitoring	: CALIbrate:INTerface:OFFset VMON

Set range 0...5V or 0...10V : SYSTem:INTerface:IANalog<sp><slot>,RANGE,<state><term>

Syntax : ***SAV<term>**

When a password is used, the only way to store this relevant settings into memory is to use:
Syntax : ***SAV<sp><password><term>**

Warning! this command overrides the settings already stored in memory.

*CLS

This reset command clears the error queue. See paragraph 5.7 "Error Message".

*RST

This reset command sets the power supply in a save defined state. The table below gives an overview of the settings made after sending this reset command or after power-on of the SM15K:

5.2 Source Subsystem

Introduction

Setting	Value	set after *RST:	set after power-on:
SOURce:VOLtage	0	YES	YES
SOURce:CURrent	0	YES	YES
SOURce:CURrent:NEGative	0	YES	YES
SOURce:POWer	0	YES	YES
SOURce:POWer:NEGative	0	YES	YES
SYSTem:RSD	OFF	YES	YES
SYSTem:REM:CV	REM	YES	NO
SYSTem:REM:CC	REM	YES	NO
SYSTem:REM:CP	REM	YES	NO
SYSTem:FROntpanel	OFF	YES	NO

The parameters (output voltage / output current / output power) have a working range from 0 to the maximum output voltage / current / power of the power supply. The response strings contain integers (max. values) or floating point values with 4 digits of precision (set values).

Maximum Output Voltage

To read the maximum output voltage, send the query:
Syntax : **SOURce:VOLtage:MAXimum?<term>**
For example the answer is: 500<term>

Maximum Output Current

To read the maximum output current, send the query:
Syntax : **SOURce:CURrent:MAXimum?<term>**

Maximum Output Current Negative

To read the maximum negative output current, send the query:
Syntax : **SOURce:CURrent:NEGative:MAXimum?<term>**

Maximum Output Power

To read the maximum output power, send the query:
Syntax : **SOURce:POWer:MAXimum?<term>**

Maximum Output Power Negative

To read the maximum negative output power, send the query:
Syntax : **SOURce:POWer:NEGative:MAXimum?<term>**

Set Output Voltage

To set the output voltage of the power supply:
Syntax : **SOURce:VOLtage<sp><NR2><term>**
To read the last settings, send the query:
Syntax : **SOURce:VOLtage?<term>**
For example the answer is: 14.0000<term>

Set Output Current

To set the output current of the power supply:
Syntax : **SOURce:CURrent<sp><NR2><term>**
To read the last settings, send the query:
Syntax : **SOURce:CURrent?<term>**

Set Output Current Negative

To set the output current nagative of the power supply:
Syntax : **SOURce:CURrent:NEGative<sp><NR2><term>**
To read the last settings, send the query:
Syntax : **SOURce:CURrent:NEGative?<term>**

Set Output Power

To set the output power of the power supply:
Syntax : **SOURce:POWer<sp><NR2><term>**
To read the last settings, send the query:
Syntax : **SOURce:POWer?<term>**

Set Output Power Negative

To set the output power negative of the power supply:
Syntax : **SOURce:POWer:NEGative<sp><NR2><term>**
To read the last settings, send the query:
Syntax : **SOURce:POWer:NEGative?<term>**

Output Voltage Stepsize

To read the programming stepsize of the output voltage, send the query:
Syntax : **SOURce:VOLtage:STEpsize?<term>**

The reply will be in scientific notation, with an accuracy of 15 decimals.
e.g.: 1.055924221873283e-03<term>
This represents the smallest Voltage programming step possible.

Output Current Stepsize

To read the programming stepsize of the output current, send the query:
Syntax : **SOURce:CURrent:STEpsize?<term>**

The reply will be in scientific notation, with an accuracy of 15 decimals.e.g.:
1.759365200996399e-03<term>
This represents the smallest Current programming step possible.

Output Power Stepsize

To read the programming stepsize of the output power, send the query:
Syntax : **SOURce:POWer:STEpsize?<term>**

The reply will be in scientific notation, with an accuracy of 15 decimals.e.g.:
8.935988545417786e-01<term>
This represents the smallest Power programming step possible.

5.3 Measure Subsystem

Introduction

To measure the power supply output parameters Voltage, Current and Power, three different queries are available.
These commands measure the actual output parameters, which are not necessarily the same as the output settings like SOURce:VOLtage, SOURre:CURrent and SOURce:CURrent:NEGative, SOURce:POWer and SOURce:POWer:NEGative .
If for example the output settings are 15 V and 5 A and the unit is in CC-mode (Constant Current), the output voltage is not equal to the setting of 15 V, but less.

Measure Output Voltage

To measure the output voltage of the power supply:
Syntax : **MEASure:VOLtage?<term>**
The resolution of the answer is 16 bits, displayed with 4 digits of precision.
The answer is in Volts.

Measure Output Current

To measure the output current of the power supply:
Syntax : **MEASure:CURrent?<term>**
The resolution of the answer is 16 bits, displayed with 4 digits of precision.
The answer is in Amps.

Measure Output Power

To measure the output power of the power supply, send this query.
Syntax : **MEASure:POWer?<term>**
The resolution of the answer is 16 bits, displayed with 2 digits of precision.
The answer is in Watts.

Instrument

To measure Ampere Hour (Ah), Watt Hour (Wh), minimum/maximum output current or power, an internal instrument can be enabled.
The sampling interval is 100ms.

Ampere Hour Instrument

To enable the current measurement instrument:
Syntax: **MEASure:INStrument<sp>AH,STATE,<boolean><term>**
(Re)enabling the instrument will reset all previous measurements.
To read the enable status:
Syntax: **MEASure:INStrument<sp>AH,STATE?<term>**

To read the time the instrument is active:
Syntax: **MEASure:INStrument<sp>AH,TIMEHR?<term>**
The result will be in hours with three decimals.
If the instrument is not enabled, the result will be zero.
Syntax: **MEASure:INStrument<sp>AH,TIMESEC?<term>**
The result will be in seconds with one decimal.
If the instrument is not enabled, the result will be zero.

To read the total Ampere Hour for the positive current:
Syntax: **MEASure:INStrument<sp>AH,POS,TOTAL?<term>**
The result will be in Ampere Hour (Ah), scientific notation. If the instrument is not enabled, the result will be zero.

To read the total Ampere Hour for the negative current:
Syntax: **MEASure:INStrument<sp>AH,NEG,TOTAL?<term>**
The result will be in Ampere Hour (Ah), scientific notation.
If the instrument is not enabled, the result will be zero.

To read the minimum positive current during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>AH,POS,IMIN?<term>**
The result will be in Ampere. If the instrument is not enabled, the result will be zero.

To read the maximum positive current during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>AH,POS,IMAX?<term>**
The result will be in Ampere. If the instrument is not enabled, the result will be zero.

To read the minimum negative current during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>AH,NEG,IMIN?<term>**
The result will be in Ampere. If the instrument is not enabled, the result will be zero.

To read the maximum negative current during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>AH:NEG,IMAX?<term>**
The result will be in Ampere. If the instrument is not enabled, the result will be zero.

Watt Hour Instrument

To enable the power measurement instrument:
Syntax: **MEASure:INStrument<sp>WH,STATE,<boolean><term>**
Enabling the instrument will reset all previous measurements to zero.
To read the enable status:
Syntax: **MEASure:INStrument<sp>WH,STATE?<term>**

To read the time the instrument is active:
Syntax: **MEASure:INStrument<sp>WH,TIMEHR?<term>**
The result will be in hours with three decimals.
If the instrument is not enabled, the result will be zero.
Syntax: **MEASure:INStrument<sp>WH,TIMESEC?<term>**
The result will be in seconds with one decimal.
If the instrument is not enabled, the result will be zero.

To read the total Watt Hour for the positive power:
Syntax: **MEASure:INStrument<sp>WH,POS,TOTAL?<term>**
The result will be in Watt Hour (Wh), scientific notation.
If the instrument is not enabled, the result will be zero.

To read the total Watt Hour for the negative power:
Syntax: **MEASure:INStrument<sp>WH,NEG,TOTAL?<term>**
The result will be in Watt Hour (Ah), scientific notation.
If the instrument is not enabled, the result will be zero.

To read the minimum positive power during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>WH,POS,PMIN?<term>**
The result will be in Watts. If the instrument is not enabled, the result will be zero.

To read the maximum positive power during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>WH,POS,PMAX?<term>**
The result will be in Watts. If the instrument is not enabled, the result will be zero.

To read the minimum negative power during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>WH,NEG,PMIN?<term>**
The result will be in Watts. If the instrument is not enabled, the result will be zero.

To read the maximum negative power during the time the instrument is enabled:
Syntax: **MEASure:INStrument<sp>WH:NEG,PMAX?<term>**
The result will be in Watts. If the instrument is not enabled, the result will be zero.

Measure temperature

To read the highest internal temperature of the power supply:
Syntax: **MEASure:TEMperature?<term>**
The answer is in degrees Celsius.

5.4 Calibrate Subsystem

Introduction

The calibration of the power supply is done during production. However, periodical check and calibration is recommended. At power-on, the calibration settings are restored.
There are four parameters to calibrate. Two related to the voltage measurement and two related to the current measurement.

For proper calibration it is recommended to use the following order (an SM500-CP-90 is used as example) :
- Set output voltage of the power supply to e.g. 1% of the maximum output voltage.
  SOURce:VOLtage<sp>5.00<term>
- Calibrate the measure voltage offset, so the result of the command MEASure:VOLtage? is as close as possible to 5.00 V.
- Set the output voltage to maximum
  SOURce:VOLtage<sp>500<term>
- Calibrate the measure voltage gain, so the result of the command MEASure:VOLtage? is as close as possible to the actual output voltage, 500 V.

Same principle is used for the current calibration.

Default settings for the offsets are 0, and default settings for the gains are 1. The resolution of both settings and readings are 6 digits.
Notice: Both gain and offset changes influence the actual output parameter. After changing offset, check if the gain has to change. And visa versa. To get a very accurate result, redo the calibration a few times. To save the calibration settings to the non-volatile memory, refer to section 5.1

For MEASURE offset calibration, use the equation:

$$new_{offset} = old_{offset} + actualValue - programmedValue$$

For MEASURE gain calibration, use the equation:

$$new_{gain} = old_{gain} * \left(\frac{actualValue}{programmedValue} \right)$$

Calibrate Gain Measure Current

To calibrate the gain of the current measurement:
Syntax : **CALIbrate:CURrent:MEAsure:GAIn<sp><NR2><term>**
To read the calibration settings:
Syntax : **CALIbrate:CURrent:MEAsure:GAIn?<term>**

Calibrate Gain Measure Voltage

To calibrate the gain of the voltage measurement:
Syntax : **CALIbrate:VOLtage:MEAsure:GAIn<sp><NR2><term>**
To read the calibration settings:
Syntax : **CALIbrate:VOLtage:MEAsure:GAIn?<term>**

Calibrate Offset Measure Current

To calibrate the offset of the current measurement:
Syntax : **CALIbrate:CURrent:MEAsure:OFFset<sp><NR2><term>**
To read the calibration settings:
Syntax : **CALIbrate:CURrent:MEAsure:OFFset?<term>**

Calibrate Offset Measure Voltage

To calibrate the offset of the voltage measurement:
Syntax : **CALIbrate:VOLtage:MEAsure:OFFset<sp><NR2><term>**
To read the calibration settings:
Syntax : **CALIbrate:VOLtage:MEAsure:OFFset?<term>**

Note: Range Offset : About $1/_{30}$ of the maximum output voltage/current, Range Gain : 0.9 to 1.1

5.5 Power Sink

Introduction

The following commands are available to read and change the Current Permission settings of the Power Sink. Make sure to set the negative current as well to actually feedback power into the grid.

RSD

To set up the Power Sink that it reacts on RSD:
Syntax: **SYSTem:POWersink<sp>rsd,<boolean><term>**
To check the setting:
Syntax: **SYSTem:POWersink<sp>rsd?<term>** 1 = reacts on RSD

Interlock

To set up the Power Sink that it reacts on Interlock:
Syntax: **SYSTem:POWersink<sp>interlock,<boolean><term>**
To check the setting:
Syntax: **SYSTem:POWersink<sp>interlock?<term>** 1 = reacts on Interlock

Output On/Off

To set up the Power Sink that it reacts on Output On/Off:
Syntax: **SYSTem:POWersink<sp>output,<boolean><term>**
To check the setting:
Syntax: **SYSTem:POWersink<sp>output?<term>** 1 = reacts on Output On/Off

5.6 System Subsystem

Remote Shut Down (RSD)

Syntax : **SYSTem:RSD[:STAtus]<sp><boolean><term>**
Boolean = 0, 1, OFF (Unlocked), ON (Locked)
To read the last setting:
Syntax : **SYSTem:RSD[:STAtus]?<term>**

Limits

To set the limits of the voltage:
Syntax : **SYSTem:LIMits:VOLtage<sp><NR2>,<boolean><term>** Off = disabled, On = enabled
To read the last setting:
Syntax : **SYSTem:LIMits:VOLtage?**

To set the limits of the current:
Syntax : **SYSTem:LIMits:CURrent<sp><NR2>,<boolean><term>** Off = disabled, On = enabled
To read the last setting:
Syntax : **SYSTem:LIMits:CURrent?**
To set the limits of the negative current:
Syntax : **SYSTem:LIMits:CURrent:NEGative<sp><NR2>,<boolean><term>** Off = disabled, On = enabled
To read the last setting:
Syntax : **SYSTem:LIMits:CURrent:NEGative?**

To set the limits of the power:
Syntax : **SYSTem:LIMits:POWer<sp><NR2>,<boolean><term>** Off = disabled, On = enabled
To read the last setting:
Syntax : **SYSTem:LIMits:POWer?**

To set the limits of the negative power:
Syntax : **SYSTem:LIMits:POWer:NEGative<sp><NR2>,<boolean><term>** Off = disabled, On = enabled
To read the last setting:
Syntax : **SYSTem:LIMits:POWer:NEGative?**

Front Panel Highlight

Syntax : **SYSTem:FROntpanel:HIGhlight**
- Display on front will blink for about 2 seconds.
- Buzzer on front is on for about 2 seconds.

Front Panel Lock

It is possible to lock either the Menu, or lock the Menu and the Controls.
Syntax : **SYSTem:FROntpanel[:STAtus]<sp><boolean><term>**
Boolean = 0, 1, OFF (Unlocked), ON (Locked)
To read the last setting:
Syntax : **SYSTem:FROntpanel[:STAtus]?<term>**

Both the Menu and the Controls can be locked. To set whether the Menu, or the Menu and the Controls will be locked use the command:
Syntax : **SYSTem:FROntpanel:CONtrols<sp><boolean><term>**
To read the last setting:
Syntax : **SYSTem:FROntpanel:CONtrols?**

Boolean = 0 (Menu), 1 (Menu & Controls), OFF (Menu), ON (Menu & Controls)

Remote method

The SM15K allows the user to switch to either local or remote programming.
The programming of CV, CC or CP can be done individually, so for example one or two can be controlled via Ethernet interface and the others via the encoder on the frontpanel. Any combination is possible. Hence there are three commands to set the programming source:

To set the CV programming source:
Syntax : **SYSTem:REMote:CV[:STAtus]<sp><setting><term>**setting = Remote or Local[1]
To read the last setting:
Syntax : **SYSTem:REMote:CV[:STAtus]?<term>**

To set the CC programming source:
Syntax : **SYSTem:REMote:CC[:STAtus]<sp><setting><term>**setting = Remote or Local[1]
To read the last setting:
Syntax : **SYSTem:REMote:CC[:STATus]?<term>**

To set the CP programming source:
Syntax : **SYSTem:REMote:CP[:STAtus]<sp><setting><term>**setting = Remote or Local[1]
To read the last setting:
Syntax : **SYSTem:REMote:CP[:STATus]?<term>**
1) The settings Remote or Local are equal to the settings Ethernet or Front.

Possible settings are :
- Front (Local)
- Web
- Sequencer

- Ethernet (Remote)
- Slot1 - Slot4
- Function1

Time and date

By default the power supply has no time and date set. The user can set these by using the following commands:

Time:
Syntax: **SYSTem:TIMe<sp><hour>,<minute>,<second><term>**

Hour = 0-23
Minute = 0-59
Second = 0-59

To read the current time:
Syntax: **SYSTem:TIMe?<term>**
Answer = <hour>:<minute>:<second>

The answer will be “UNKNOWN” in case the time was not set.

Date:
Syntax: **SYSTem:DATe<sp><year>,<month>,<day><term>**

Year = 2019-2099
Month = 1-12
Day = 1-31

To read the current date:
Syntax: **SYSTem:DATe?<term>**
Answer = <year>-<month>-<day>

The answer will be “UNKNOWN” in case the date was not set. Both time and date are volatile and not restored after a power cycle.

Error Message

If an unknown command, an invalid value or an illegal setting is received by the SM15K an error is generated. The user is prompted by the Error LED on the RJ45 connector and an error message is stored in the error queue, which contains the error number and a description of the kind of error.

The error queue can contain maximum 10 errors; more error messages are ignored. The command to read the queue line by line is:
Syntax : **SYSTem:ERRor?<term>**
The SM15K returns the first error and clears it from the queue. If there are no errors (so the queue is empty), the result of this query will be : **0,None<term>**
So after 10 readings of SYSTem:ERRor? the queue is empty for sure, or after using the *CLS command.

Warning messages

If there are issues that generated a warning, it can be read by the following command:
Syntax: **SYSTem:WARning?<term>**
If there are no warnings, the result of this query will be: 0,None<term>

Password

To protect the most essential settings of the system (calibration values, *PUD, password) a password can be used. The default password is : "**DEPOWER**".
To apply a password, send the command:
Syntax : **SYSTem:PASsword<sp><old_password>,<new_password><term>**

If no password is used, <old_password> must be **"DEFAULT"** (case independent) and <new_password> the password to be used.
To remove the password, <old_password> must be the current password and <new_password> must be **"DEFAULT"** (case independent).
Maximum length of password is 9 characters.

Note: When a password is unknown or forgotten, the reset switch can be pressed for four

seconds. This will restore the factory default password. Note that the Network settings will also be set to factory default "DHCP Enable"
In the SM15K manual, see paragraph 8.10 for a detailed description about the reset switch.

To store the new password, refer to section 5.1(*SAV)

To read if a password is used, send the query
Syntax : **SYSTem:PASsword:STAtus?<term>**
If no password is used, the SM15K will return 0<term>, otherwise it returns 1<term>.

Watchdog

The SM15K provides a Watchdog timer on the Ethernet interface. The power supply monitors the Ethernet communication when set and disables its power output when no Ethernet command is received within the time set.

The Watchdog timer is disabled after a power-on event. Send the set command once to activate the Watchdog timer and reset the timer by sending any valid Ethernet command at regular intervals.

To set the Watchdog timer (in ms):
Syntax : **SYSTem:COMmunicate:WATchdog<sp>SET,<NR1><term> <NR1>= 20…10000**

To read the last setting: (valid until Timeout)
Syntax : **SYSTem:COMmunicate:WATchdog<sp>SET?<term>**

To read the current state of the Watchdog timer:
Syntax : **SYSTem:COMmunicate:WATchdog?<term>**

There are three possibilities:

20…10000<term>	Current timer value in ms
0<term>	Timeout. Clears indicator on Front panel and Web
-1<term>	Clears Timeout, Watchdog is off

To disable the Watchdog timer:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>STOP<term>**

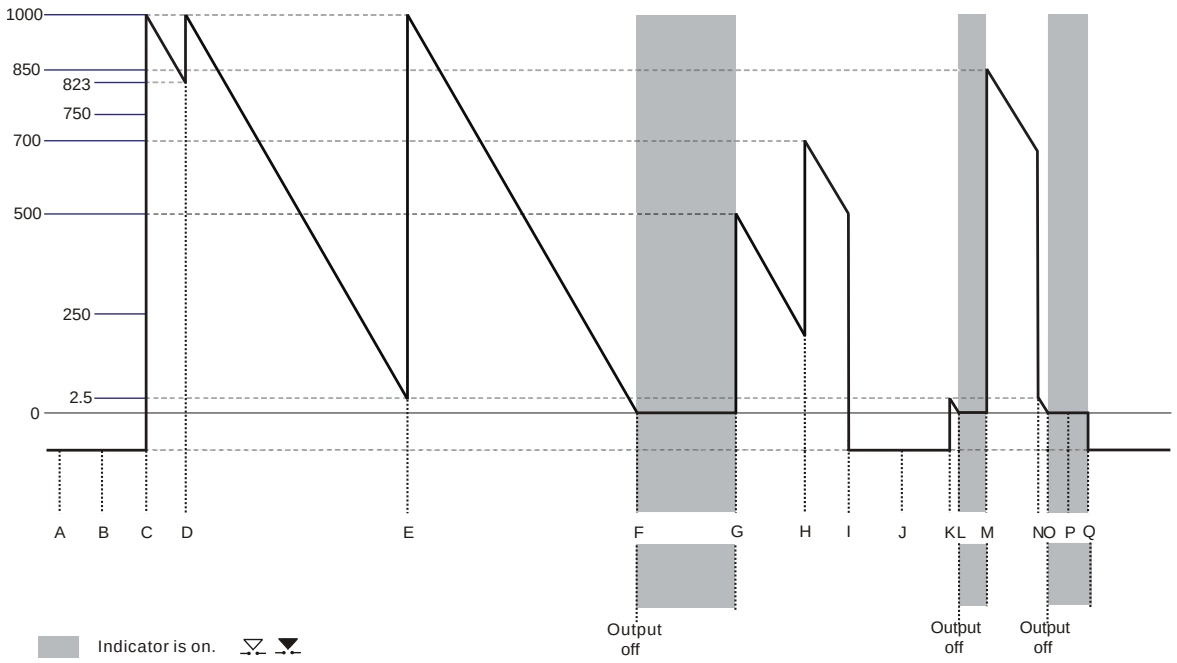
To test the Watchdog timer:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>TEST<term>**

The timer will be set to 2.5ms and expires very quickly. The indicator on the Front panel and Web will be activated. Enable, disable or query the state of the Watchdog timer to clear the indicators.

Watchdog Example

A = Power on event, communication with the watchdog is off.
B = Gives -1, Indicates that the watchdog is still off.
Syntax : **SYSTem:COMmunicate:WATchdog?<term>**
C = This will set the times of the watchdog to 1000 ms, and enables it.
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set,1000<term>**
D = This indicates that the watchdog timer was at 823 ms when the command was received.
Syntax : **SYSTem:COMmunicate:WATchdog?<term>** gives 823
To reset the watchdog timer, use the following command:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set,1000<term>**
To see in which period value of the watchdog timer is, use the following command,
in this case it is 1000 ms.
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set?<term>**
E = Any valid command will reset the watchdog timer.
F = The watchdog timer expires and switches the output off.
G = To set the watchdog timer to 500 ms, use the command:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set,500<term>**
H = To set the watchdog timer to 700 ms, use the command:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set,700<term>**

I = To disable the watchdog timer use the command:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>stop<term>**
J = Gives -1, Indicates that the watchdog is still off.
Syntax : **SYSTem:COMmunicate:WATchdog?<term>**



K = To test the watchdog timer use the following command, it loads the watchdog timer with 2.5 ms, so it expires very quickly:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>test<term>**
L = The watchdog timer expires and switches the output off.
M = To set the watchdog timer to 850 ms, use the command:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>set,850<term>**
N = To test the watchdog timer, use the following command, it loads the watchdog timer with 2.5 ms, so it expires very quickly:
Syntax : **SYSTem:COMmunicate:WATchdog<sp>test<term>**
O = The watchdog timer expires and switches the output off.
Syntax : **SYSTem:COMmunicate:WATchdog?<term>**
P = Gives 0, This will clear the watchdog timeout indicator on the front and webserver.
Syntax : **SYSTem:COMmunicate:WATchdog?<term>**
Q = Gives -1, Indicates that the watchdog is still off and makes the count -1.

Terminator

The terminator can be chosen for the communication. The terminator is however set to default on a power-on event (ASCII character 0AH, 10d)

To set the Terminator:
Syntax : **SYSTem:COMmunicate:TERminator<sp> <value> <term>** value = CR, CRLF or LF
LF = Linefeed = 0AH = 10d
CR = Carriage Return = 0DH = 13d

To read the last setting:
Syntax : **SYSTem:COMmunicate:TERminator?<term>**

5.7 Output

The SM15K provides a command to switch the output of a power supply ON or OFF.
Syntax : **OUTPut<sp><boolean><term>** boolean = 0, 1, OFF, ON
To read the last settings:
Syntax : **OUTPut?<term>**

5.8 Register Structure

The SM15K provides two register structures which contain actual Power Supply status information.
To read the registers :
Syntax : **STATus:REGister:A?<term>**

Syntax : **STATus:REGister:B?<term>**

The SM15K will return a decimal number which represents the binary status of the status signals. For example, if the power supply is in CC-mode and signals DC-Fail, the register A condition will be : 66<term>. (= 2 + 64).

See below image for an overview of the Registers:

Status Register : A

Bit field	Bit weight	Signal
0	1	CV
1	2	CC
2	4	CP
3	8	V_{lim}
4	16	I_{lim}
5	32	P_{lim}
6	64	DCF
7	128	Reserved
8	256	OT
9	512	Reserved
10	1024	ACF
11	2048	Interlock
12	4096	RSD
13	8192	Output
14	16384	Frontpanellock
15	32768	Reserved

Status Register : B

Bit field	Bit weight	Signal
0	1	Rem CV
1	2	Rem CC
2	4	Rem CP
3	8	Program running
4	16	Wait for Trigger
5	32	M/S enab.AND Master
6	64	M/S enab.AND Slave
7	128	V_{output} overload
8	256	I_{output} overload
9	512	Reserved
10	1024	Vprg overload
11	2048	Iprg overload
12	4096	Reserved
13	8192	Reserved
14	16384	Reserved
15	32768	Program Open End Error (cleared after read)

5.9 Functions

To select a function:
Syntax: **SYSTem:FUNCtion:SELect<sp><nr1><term>**
nr1 = 1

To read which function is selected:
Syntax: **SYSTem:FUNCtion:SELect?<term>**

To configure the type of the selected function:
Syntax: **SYSTem:FUNCtion:TYPe<sp><select><term>**
Select = 0 or None,1 or Ri, 2 or LeadlessSense
Note: All configurations and settings will be default when selecting a different type.

To read the type of the selected function:
Syntax: **SYSTem:FUNCtion:TYPe?<term>**
The reply will be 0, NONE or 1,RI or 2,LEADLESSSENSE

To configure the input source of the selected function:
Syntax: **SYSTem:FUNCtion:CONfig<sp>SV,<setting><term>**
Setting = Any of the sources shown in chapter “System remote method”, except the function itself.
Note: The output will be switched off when this command is send.

To read the configured input source of the selected function:
Syntax: **SYSTem:FUNCtion:CONfig<sp>SV?<term>**

To set the parameters of the function:
Syntax: **SYSTem:FUNCtion:SETting<sp><item>,<value><term>**
If the function is configured as Ri:
Item = Ri : value = 0.0 … 10.0 (in Ohms)
Item =Vhigh : value = 0.0 … Vmax
Item =Vlow : value = 0.0 … -Vmax

If the function is configured as LeadlessSense:
Item = Rcon : value = 0.0 … 10.0 (in Ohms)
Item =Vhigh : value = 0.0 … Vmax
Item =Vlow : value = 0.0 … -Vmax

To read the parameters of the function:
Syntax: **SYSTem:FUNCtion:SETting<sp><item><term>**

If the function is configured as Ri:
Item can be: RI?, Vhigh?, Vlow?

If the function is configured as LeadlessSense:
Item can be: Rcon?, Vhigh?, Vlow?

5.10 Logging

Commands received by the unit and replies send back by the unit can be logged. This can be valuable when debugging or monitoring the Ethernet communication between software applications and the unit. This capturing can be configured to be on receive (Rx) only, transmit (Tx) only, or both.

The log file can be read via an Ethernet command and via the web page. The file can be cleared, the size can be read and the maximum file size can be queried.

Furthermore, the commands used to control and read the logging can be excluded from this logging file. This is useful when, for example, the size of the log file is monitored at a certain interval to see if the maximum file size has been reached. Setting the exclusion to ON will prevent the log file to grow while controlling or reading this Ethernet logging mechanism.

To configure the logging for received commands:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,RX,<boolean><term>**
boolean = 0,1,OFF or ON. Default = OFF.

To read the configuration for received commands:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,RX?><term>**

To configure the logging for transmitted replies:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,TX,<boolean><term>**
boolean = 0,1,OFF or ON. Default = OFF.

To read the configuration for transmitted replies:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,TX?><term>**

To configure the exclusion of the logging commands from the log file:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,EXCLUDE,<boolean><term>**
boolean = 0,1,OFF or ON. Default = OFF.

To read the configuration for exclusion:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH, EXCLUDE?><term>**

To read the log file:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,READ?><term>**
The replied log file is encapsulated between the characters '<' and '>'. Finally, the reply is ended with the terminator.

For example:
Both Rx and Tx logging are enabled, the logging commands are excluded, the terminator was set to LF and the query "MEASure:VOLtage?\n" is send. The reply on "SYSTem:COMmunicate:LOGging<sp>ETH,READ?\n" will be:

"<MEASURE:VOLTAGE?\n1.1252\n>\n".

Note that the output of the read command will not be logged. (Otherwise the size of the file would be doubled every read).

For reading the log file via the web page, please refer to Product Manual, chapter 7 - web interface, administration.

To clear the log file:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,CLEAR><term>**
Note: after sending this clear-command, the settings of RX, TX and EXCLUDE are set to default.

To read the maximum size of the log file:
Syntax: SYSTem:COMmunicate:LOGging<sp>ETH,SIZEMAX?><term>
The reply will be the maximum amount of characters in kilobytes.

To read the current size of the log file:
Syntax: **SYSTem:COMmunicate:LOGging<sp>ETH,SIZE?><term>**
The reply will be the amount of logged characters in kilobytes.

5.11 Interfaces

Introduction

Up to a number of 4 interfaces can be plugged in the sockets at the rear side of the unit.
All of these interfaces can easily be plugged in afterwards at the customer site.
The following types are available:

- Serial, USB and differential programming.
- Digital User I/O for programming.
- Floating Contacts, floating Interlock and floating Enable.
- Master Slave interface.
- Isolated analog programming & monitoring, logic status outputs

Installed interfaces

To read which type of interface is installed for a specific slot, send the query:
SYSTem:INTerface:TYPe<sp><slot>?

To read the type of installed interface of all slots send the query:
SYSTem:INTerface:TYPe<sp>ALL?

The answer is separated by a semicolon, respectively for slot 1, 2, 3 and 4.
For example the answer is: **DigIO;Serial;IsoAna;Ms2**

5.11.1 Digital User In-/Outputs (optional)

The optional Interface Digital I/O provides 8 user inputs and 8 users outputs. These can be controlled / monitored by commands (explained below) or can be used to interact with the Sequencer (refer to chapter 6). A total of 4 Digital I/O Interfaces can be inserted and used.

The user inputs and user outputs are floating (maximum $60V_{DC}$) from earth.

User Outputs

To program the user outputs, a decimal number can be send to the SM15K. This decimal number represents the binary state of the 8 user outputs.

Syntax : **SYSTem:INTerface:DIO:OUTput<sp><slot>,<0+NR1><term>** <0+NR1>= 0,1,2,3.....255
Output A = 1
Output B = 2
Output C = 4
Output D = 8
Output E = 16
Output F = 32
Output G = 64
Output H = 128

For example, to set output C and H of the Digital I/O Interface inserted in slot 1 send:
SYSTem:INTerface:DIO:OUTput<sp>1,132<term>
To reset all the user outputs of a specific Digital I/O Interface, send:
SYSTem:INTerface:DIO:OUTput<sp><slot>,0<term> (default setting after power-on).

To read the last setting of the user outputs of specific Digital I/O Interface:
Syntax : **SYSTem:INTerface:DIO:OUTput<sp><slot>?**

To read the last settings of all inserted Digital I/O Interfaces:
Syntax : **SYSTem:INTerface:DIO:OUTput<sp>ALL?**

User Inputs

To read the status of the 8 user inputs, the User Input Condition Register can be read:
Syntax : **SYSTem:INTerface:DIO:INPut<sp><slot>?**
The SM15K will return a decimal number, which represents the binary status of the 8 inputs.

Input A = 1
Input B = 2
Input C = 4
Input D = 8
Input E = 16
Input F = 32
Input G = 64
Input H = 128

For example, if only Input A and Input G are high, the condition will be : **65<term>**. (=1+64)

To read the status of all insterted Digital I/O Interfaces :
Syntax : **SYSTem:INTerface:DIO:INPut<sp>ALL?**

Order code Digital I/O interface : INT MOD DIG

5.11.2 Isolated Contacts (optional)

The optional Interface Isolated Contacts provides 4 changeover Relay contacts, an Interlock (additional to the standard SM15K Interlock) and an Enable Input. The Relay contacts can be controlled / monitored by commands (explained below). The Interlock and Enable Input can be monitored by commands (explained below). A total of 4 Isolated Contacts Interfaces can be inserted and used.

The Relay Contacts, the Interlock and the Enable Input are floating (maximum $60V_{DC}$) from earth.

Relay Contacts

To control the relays, decimal figures can be send to the SM15K representing the slot number, relay number and relay value.
Note: these commands will only work if Relay-Status-Linkage is not used.

To set a specific Relay:
Syntax : **SYSTem:INTerface:ICOntacts:RELay<sp><slot>,<relay>,<value><term>**
slot = 1 - 4, relay = 1 - 4, value = 0 or 1

To read the status of a specific Relay:
Syntax : **SYSTem:INTerface:ICOntacts:RELay<sp><slot>,<relay>?<term>**
To read the status of all Relays in a specific slot:
Syntax : **SYSTem:INTerface:ICOntacts:RELay<sp><slot>?<term>**

To read the Relay status of all inserted Interfaces Isolated Contacts:
Syntax : **SYSTem:INTerface:ICOntacts:RELay<sp>ALL?<term>**

Relay-Status-Linkage

Several system statuses can be linked to a specific relay. If linkage is used, the Relay Contacts commands will not function anymore.

To link a specific Relay to a system status:
Syntax : **SYSTem:INTerface:ICOntacts:LINkrelay<sp><slot>,<relay>,<value><term>**
slot=1 - 4, relay=1 - 4, value = ACF, DCF, INTERLOCK, OUTPUT, RSD, LIMIT, OT or DEFAULT. (DEFAULT makes the Relay Contacts commands work again)

To read which system status is linked to a specific Relay:
Syntax : **SYSTem:INTerface:ICOntacts:LINkrelay<sp><slot>,<relay>?<term>**
slot = 1 - 4, relay = 1 - 4

To read which system statuses are linked to a specific slot:
Syntax : **SYSTem:INTerface:ICOntacts: LINkrelay<sp><slot>?<term>**

Programmed linkage settings will be restored at power-up if the INT MOD CON remains in the same slot.

Interlock

The Interlock is functionally parallel to the Interlock standard available in the SM15K and therefore they both need to have their own input pins (pin 1 and pin 3) linked for operation. The difference is that the additional Interlock is floating (maximum $60V_{DC}$) from Earth, while the standard Interlock is referenced to Earth.

To read the status of an Interlock in a particular slot:
Syntax : **SYSTem:INTerface:ICOntacts:INTerlock<sp><slot>?<term>**

To read the Interlock status of all inserted Interfaces Isolated Contacts:
Syntax : **SYSTem:INTerface:ICOntacts:INTerlock<sp>all?<term>**

Enable Input

The Enable input (pin 2) can be used to Enable the power supply output using a $24V_{DC}$ signal from for example a PLC. Use pin 3 as Return. Remove the additional Interlock link when using the Enable input.

To read the status of an Enable pin in a particular slot:
Syntax : **SYSTem:INTerface:ICOntacts:ENAble<sp><slot>?<term>**

To read the Enable status of all inserted Interfaces Isolated Contacts:
Syntax : **SYSTem:INTerface:ICOntacts:ENAble<sp>all?<term>**

Order code Isolated Contacts interface : INT MOD CON.

5.11.3 Isolated Analog (optional)

The optional Interface Isolated Analog provides:
- 2 analog programming inputs
- 2 analog monitoring outputs
- 6 digital status outputs
- Remote shutdown input
- Reference voltage of 5.1V
- 12V auxiliary output.

The analog inputs and outputs can be calibrated or configured for 0-5V or 0-10V range by the commands explained below. Maximum 1 Isolated Analog interface can be used in a unit.

Analog input and output range

The selection of the range 0-5V or 0-10V applies to the inputs and outputs simultaneously.
Syntax: **SYSTem:INTerface:IANalog<sp><slot>,RANGE,<state><term>**
State = 0, LO (for 0-5V range) or 1, HI (for 0-10V range)
To read the range:
Syntax: **SYSTem:INTerface:IANalog<sp><slot>,RANGE?<term>**

Important:
If calibration was done without saving to non-volatile memory first (see section 5.1), switching to another range will restore the previous calibration values.

Calibration introduction:

The calibration of the INT MOD ANA is done during production. However, periodical check and calibration is recommended. At power-on, the calibration settings are restored. There are eight parameters to calibrate. Four related to the voltage programming / monitoring and four related to the current programming / monitoring.

For proper calibration it is recommended to use the following order (an SM500-CP-90 + INT MOD ANA set to 5V range is used as example) :
- Set output voltage of the power supply to e.g. 1% of the maximum output voltage by applying 50 mV.
- Calibrate the programming voltage offset, so the output voltage is as close as possible to 5.0 V.
- Calibrate the monitor voltage offset, so the voltage on the Vmonitor output is as close as possible to 50 mV.
- Set the output voltage to maximum by applying 5V.
- Calibrate the programming voltage gain, so the output voltage is as close as possible to the maximum voltage, 500V.
- Calibrate the monitor voltage gain, so the voltage on the Vmonitor output is as close as possible to 5V.

Same principle is used for the current calibration.

Default settings for the offsets are 0, and default settings for the gains are 1. The resolution of both settings and readings are 4 digits.

Note: Both gain and offset changes influence the actual output parameter. After changing offset, check if the gain has to change. And visa versa. To get a very accurate result, redo the calibration a few times.

To save the calibration settings to the non-volatile memory, refer to section 5.1

Since the programming and monitoring is proportional to the output voltage / current, scaling should be taken into account.
Example:
Situation: SM500-CP-90 with INT MOD ANA is configured for analog programming with 0-5V range.

Programming voltage is 5.001V, Vprg gain is 1.003 and the output voltage results in 499.75V
New Vprg gain is:

Programmed voltage / actual voltage * old gain = 500.100 / 499.705 * 1.003 = 1.0038
Note: the programmed voltage is not 5.001V, but the expected output voltage of the unit itself.

Note: Range offset: About $^{1}/_{33}$ of the maximum voltage / current,
Range gain: 0.9 to 1.1.

For PROGRAMMING offset voltage calibration, use the equation:

$$new_{offset} = old_{offset} + programmedValue - actualValue$$

For PROGRAMMING offset current calibration, use the equation:

$$new_{offset} = old_{offset} + actualValue - programmedValue$$

For PROGRAMMING gain calibration, use the equation:

$$new_{gain} = old_{gain} * \left(\frac{programmedValue}{actualValue} \right)$$

For MONITORING offset calibration, use the equation:

$$new_{offset} = old_{offset} + actualValue - measuredValue$$

For MONITORING gain calibration, use the equation:

$$new_{gain} = old_{gain} * \left(\frac{actualValue}{measuredValue} \right)$$

Calibrate Gain Current Programming

To calibrate the analog programming gain of the current setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,IPRG,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,IPRG?<term>**

Calibrate Offset Current Programming

To calibrate the analog programming offset of the current setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,IPRG,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,IPRG?<term>**

Calibrate Gain Voltage Programming

To calibrate the analog programming gain of the voltage setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,VPRG,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,VPRG?<term>**

Calibrate Offset Voltage Programming

To calibrate the analog programming offset of the voltage setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,VPRG,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,VPRG?<term>**

Calibrate Gain Current Monitoring

To calibrate the analog programming gain of the current monitoring:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,IMON,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,IMON?<term>**

Calibrate Offset Current Monitoring

To calibrate the analog programming offset of the current monitoring:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,IMON,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,IMON?<term>**

Calibrate Gain Voltage Monitoring

To calibrate the analog programming gain of the voltage monitoring:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,VMON,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:GAIn<sp><slot>,VMON?<term>**

Calibrate Offset Voltage Monitoring

To calibrate the analog programming offset of the voltage monitoring:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,VMON,<NR2><term>**
To read the calibration setting:
Syntax: **CALIbrate:INTerface:OFFset<sp><slot>,VMON?<term>**

5.11.4 Master / Slave Interface (optional)

The optional Master Slave interface provides the ability to create larger systems with multiple power supplies. The resulting combination of units, where each unit is equipped with one master slave interface, behaves like one power supply and can be controlled or programmed on the master. A maximum of 1 master slave interface can be inserted and used per power supply. In master/slave mode, the sequencer is still available and can control the entire m/s system.

Settings

Master Slave State

To configure the state (off, slave or master) of one unit, send the command:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>STATE,<value><term>**
Value = OFF, SLAVE, MASTER or 0, 1, 2 respectively.

To read the current state, send the query:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>STATE?<term>**
Answer = “0,Off”, “1,Slave” or “2,Master”

Units in parallel

To configure the number of units in parallel, send the following command:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>PAR,<value><term>**
Value = 1 – 20

To read the number of units configured in parallel, send the query:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>PAR?<term>**

Units in series

To configure the number of units in series, send the following command:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>SER,<value><term>**
Value = 1 – 6

To read the number of units configured in series, send the query:
Syntax: **SYSTem:INTerface:MASterslave:SETting<sp>SER?<term>**

Status

ID

Each unit in a master slave system has a unique ID number. To read this ID number send the query:
Syntax: **SYSTem:INTerface:MASterslave:STAtus<sp>ID?<term>**

Configuration

To read the configuration status of one unit, send the query:
Syntax: **SYSTem:INTerface:MASterslave:STAtus<sp>CONFIG?<term>**
Answer = “pending” for when the system is configuring , “done” for when the system is ready.

Units

To read the number of units that are detected, including the master, send the query:
Syntax: **SYSTem:INTerface:MASterslave:STAtus<sp>UNITS?<term>**

Error

To read an error from the master slave system, send the query:
Syntax: **SYSTem:INTerface:MASterslave:STAtus<sp>ERROR?<term>**
Answer = "none", "external communication error" or "internal communication error".

6 Sequencer

6.1 Introduction

The SM15K includes a subsystem called SEQUENCER. This system can contain max 25 free programmable sequences of 2000 steps each. Sequences are identified by name (max 16 characters, case insensitive).
Sequences can be started and stopped by commands (see section 6.3) or by a user input (see section 6.5). That makes it possible to run stand-alone, so no PC or network is required.

A sequence can set the output voltage / current / power, set or clear digital I/O, make (un)conditional jumps, etc. It allows the user to generate arbitrary waveforms, interact with I/O like sensors, valves, motors, etc. It can also contain subroutines.
In short : it behaves like a PLC, without the bother of an extra rack unit or interconnections.

Sequences are built with steps, see the examples in section 6.6. Each step contains a step number, followed by a command with its required operand(s). Step numbers and commands are separated by a <sp> (space).
For example : 12<sp>SV=5
The execution time of a step is approximately 125µs.

The full functionality of the sequencer is also available in M/S systems.

6.2 Commands

The commands available within a sequence are sorted in categories, such as Settings, Jumps, Arithmetic and Miscellaneous. Next paragraphs describe the syntax : the commands and their operands.

Settings

- **SV**

SV stands for Source Volt. This command sets the output voltage of the power supply.
Syntax : **SV=<NR2>** <NR2> = 0 to Vmax

- **SC**

SC stands for Source Current. This command sets the output current of the power supply.
Syntax : **SC=<NR2>** <NR2> = 0 to Cmax

- **SP**

SP stands for Source Power. This command sets the output power of the power supply.
Syntax : **SP=<NR2>** <NR2> = 0 to Pmax

- **SCN**

SCN stands for Source Current Negative. This command sets the negative output current or the so-called sink current of the power supply.
Syntax : **SCN=<NR2>** <NR2> = –Cmin to 0

- **SPN**

SPN stands for Source Power Negative. This command sets the negative output power or the so-called sink power of the power supply.
Syntax : **SPN=<NR2>** <NR2> = –Pmin to 0

Note: In contrary to Front Menu Operation, for Sequencer Programming the value for SP and SPN is standard set to 0. In order to deliver or to sink power, a value other than 0 must be set.

- **Ox**

O stands for user Output. X can be A to H (output A to H). This command can set or reset a digital output.
Syntax : **Ox<slot>=<boolean>** <boolean> = 0 or 1 x = A,B,C,D,E,F,G or H
<slot> = 1,2,3 or 4

- **#x**

stands for Variable. x can be A to H. This command sets a value in a variable.
Syntax : **#x=<NR1>** <NR1> = 0 to 65535 x = A,B,C,D,E,F,G or H

- **#I**

Variable I (#I) is a timer of 1 ms. This command sets a value in the variable and the value decreases every 1 ms with 1 until it reaches zero.
Syntax : **#I=<NR1>** <NR1> = 0 to 65535

- **#J**

Variable J (#J) is a timer of 100msec. This command sets a value in the variable and the value decreases every 100msec with 1 until it reaches zero.
Syntax : **#J=<NR1>** <NR1> = 0 to 65535

Jumps

- **JP**

JP stands for Jump. It's an unconditional jump to a step anywhere in the sequence.
Syntax : **JP<sp><label>** <label> = jump to step defined by label.
See section 6.3, Add labels for a description on labels.

- **JS / RET**

JS stands for Jump to Subroutine. RET stands for Return. These two commands (always used together) allow to create subroutines within a sequence. JS <step> jumps to the subroutine located at <step>. The commands in the subroutine will be executed until a step contains RET. Then the subroutine is finished and the program returns to the step after the jump instruction JS. It's possible to nest up to 6 jumps.

Warning! do not use RET without JS or visa versa. The sequencer will be in an undefined state.

Syntax : **JS<sp><label>** <label> = jump to step defined by label.
RET
See section 6.3, Add labels for a description on labels.

- **CJE**

CJE stands for Compare Jump if Equal. Can be used in combination with Inputs, Outputs and Variables. CJE allows to create conditional jumps to any step within the sequence. It compares the first two operands and branches if their value is equal to the step declared in the third operand.
Syntax : **CJE<sp><Ix><slot>,<boolean>,<label>** x = input A,B,C,D,E,F,G or H
CJE<sp><Ox><slot>,<boolean>,<label> x = output A,B,C,D,E, F,G or H
CJE<sp><#x>,<NR1>,<label> x = variable A,B,C,D,E,F,G,H,I or J

- **CJNE**

CJNE stands for Compare Jump if Not Equal. Can be used in combination with Inputs, Outputs and Variables. CJNE allows to create conditional jumps to any step within the sequence. It compares the first two operands and branches if their value is not equal to the step declared in the third operand.
Syntax : **CJNE<sp><Ix><slot>,<boolean>,<label>** x = input A,B,C,D,E,F,G or H
CJNE<sp><Ox><slot>,<boolean>,<label> x = output A,B,C,D,E or F
CJNE<sp><#x>,<NR2>,<label> x = variable A,B,C,D,E,F,G,H,I or J

- **CJG**

CJG stands for Compare Jump if Greater. Can be used in combination with Source / Measure Voltage / Current / Power and the variables. CJG allows to create conditional jumps to any step within the sequence. It compares the first two operands and branches (if the first value is greater than the second) to the step declared in the third operand.
Syntax : **CJG<sp><SV>,<NR2>,<label>**
CJG<sp><MV>,<NR2>,<label>
CJG<sp><SC>,<NR2>,<label>
CJG<sp><MC>,<NR2>,<label>

CJG<sp><SCN>,<NR2>,<label>
CJG<sp><SP>,<NR2>,<label>
CJG<sp><MP>,<NR2>,<label>
CJG<sp><SPN>,<NR2>,<label>
CJG<sp><#x>,<NR1>,<label> x = variable A,B,C,D,E,F,G,H,I or J

SV stands for Source Volt, which is the voltage setting, **SC** stands for Source Current, **SCN** for Source Current Negative, **SP** for Source Power, **SPN** for Source Power Negative.
MV stands for Measure Volt, which is the actual output voltage, **MC** and **MP** stands for Measure Current and Measure Power.
For example CJG<sp>MV,10,voltage ; When the actual output voltage is greater than 10V, the program jumps to step define by the label voltage, otherwise it continues with the next step.

- **CJL**

CJL stands for Compare Jump if Less. Can be used in combination with Source / Measure Voltage / Current / Power and the variables. CJL allows to create conditional jumps to any step within the sequence. It compares the first two operands and branches if the first value is less than the second to the step declared in the third operand.

Syntax : **CJL<sp><SV>,<NR2>,<label>**
CJL<sp><MV>,<NR2>,<label>
CJL<sp><SC>,<NR2>,<label>
CJL<sp><MC>,<NR2>,<label>
CJL<sp><SCN>,<NR2>,<label>
CJL<sp><SP>,<NR2>,<label>
CJL<sp><SPN>,<NR2>,<label>
CJL<sp><MP>,<NR2>,<label>
CJL<sp><#x>,<NR1>,<label> x = variable A,B,C,D,E,F,G,H,I or J

For example CJL<sp>SC,10,increase ; When the current setting is less than 10A, the program jumps to step defined by the label define 'increase', otherwise it continues with the next step.

Arithmetic

- **INC**

INC stands for Increment. Can be used in combination with Voltage, Current, Power and Variables.

Syntax : **INC<sp><SV>,<NR2>**
INC<sp><SC>,<NR2>
INC<sp><SCN>,<NR2>
INC<sp><SP>,<NR2>
INC<sp><SPN>,<NR2>
INC<sp><#x>,<NR1> x = A, B,C,D,E,F,G,H,I or J

- **DEC**

DEC stands for Decrement. Can be used in combination with Voltage, Current, Power and Variables.

Syntax : **DEC<sp><SV>,<NR2>**
DEC<sp><SC>,<NR2>
DEC<sp><SCN>,<NR2>
DEC<sp><SP>,<NR2>
DEC<sp><SPN>,<NR2>
DEC<sp><#x>,<NR1> x = A, B,C,D,E,F,G,H,I or J

Miscellaneous

- **NOP**

NOP stands for No Operation. No actions are done when a step contains this command. This command can be used to arrange step numbers for future implementations or to create some small delays between commands.

Syntax : **NOP**

- **W**

W stands for Wait. Can be used to create an idle time. The operand declares the time (in seconds) the program has to wait before it continues with the next step.

Syntax : **W=<NR2>** <NR2> = 0.001 ... 65535

- **TRG**

TRG stands for Trigger. The program waits until a TRIGger:IMMediate command is received via TCP/IP. After that it continues with the next step of the sequence. This command sets the "Wait For Trigger"-bit in the Status:Register:B
Syntax : **TRG**

- **END**

END stands for End of program. When the system reads this command, the program will stop. Every program must have an END function included. (It's not necessary to have an END on the *last* step of the sequence ; e.g. after an END the program can have subroutines).
Syntax : **END**
When a sequence without END function is executed the "Program Open End Error"-bit is set in the *Status:Register:B*

6.3 Sequence control by commands

- **Read the Catalog**

The catalog contains all the programmed sequences. To read the catalog send the query:
Syntax : **PROGram:CATalog?<term>**
The SM15K returns a list of the names of all sequences, separated by linefeeds.
The end of the catalog is indicated by an extra <*nl*>.
In case of no sequences available, the SM15K only returns one <*term*>.
Example : *PROG:CAT?<term>* answer: *WAVE1<nl>PROCESS4<nl>RAMPUP<nl><term>*

- **How to create or select a Sequence**

To select an existing sequence or to create a new one, send the command:
Syntax : **PROGram:SELected:NAMe<sp><string><term>** string may contain the characters A-Z, 0-9 and + (max. 16 characters) The first character of string has to be an A-Z.

To read which sequence is selected, send the query:
Syntax : **PROGram:SELected:NAMe?<term>**
The SM15K returns the name of the selected sequence, followed by a <term>. Or, in case of no selection, only <*term*>.
Sequence names are not case-sensitive (during selection), but are stored in memory with upper case.

- **Upload a Sequence to SM15K (PC → PS)**

First select/create a sequence, then upload the new steps by the command:
Syntax : **PROGram:SELected:STEp<sp><NR1><sp><command+operand(s)><term>**
<NR1> = 1 to 2000. Steps do not have to be programmed in order, but already existing steps will be overwritten when reselected.
For example : PROG:SEL:STEp<sp>21<sp>CJNE<sp>#A,3,15<term>

- **Download a Sequence from SM15K (PS → PC)**

After a sequence is selected, the steps can be downloaded one by one, using the query:
Syntax : **PROGram:SELected:STEp<sp><NR1>?<term>**
The SM15K returns <*step number*><*sp*><*command+operand(s)*><*term*>.
If the queried step doesn't exist, the SM15K returns <*term*>.

The SM15K returns the complete sequence when it receives the query:
Syntax : **PROGram:SELected:STEp<sp>?<term>**
The answer from the SM15K will be:
<*1*><*sp*><*command+operand(s)*><*nl*><*2*><*sp*><*command+operand(s)*><*nl*> and so on.
After the last step the SM15K sends <term> as a terminator.

- **Delete a Sequence**

After a sequence is selected, it can be removed from the catalog by:
Syntax : **PROGram:SELected:DELete<term>**
To clear the whole catalog and delete all the sequences, including their assignments, send:
Syntax : **PROGram:CATalog:DELete<term>**

• Start a Sequence

If a sequence is selected, it can be started by the command:
Syntax : **PROGram:SELected:STAte<sp>RUN<term>**
The sequence will start at step 1. The RUN command will automatically initiate a sequence Build when the sequence is not yet build, or modified after an earlier build.

• Pause a Sequence

If a sequence runs, it can be paused by the command:
Syntax : **PROGram:SELected:STAte<sp>PAUSe<term>**
If the sequence is paused, it can be continued after sending the command:
Syntax : **PROGram:SELected:STAte<sp>CONTinue<term>**

• Step through a Sequence

If a sequence is selected, it is possible to manually step through the program (during any mode) by:
Syntax : **PROGram:SELected:STAte<sp>NEXT<term>**
This command executes the next step and turns in mode PAUSE. If the current step contains a Wait instruction and it is not finished yet, the sequencer ignores the Wait instruction.
For debugging the sequence this command can be very handy.
For example :
....
6 oa2=1
7 w=100
8 ob4=1
....
If the program executes step 7 to wait 100 seconds, the sequencer goes to step 8 after the SM15K received the command PROG:SEL:STA NEXT. Even when less than 100 seconds are passed.
If a step within the sequence contains a Jump instruction (CJNE,CJE, etc) which must check a certain condition to be true before the next step can be executed, an extra step before this Jump instruction is required. Otherwise the NEXT command will ignore the condition. E.g:
.....
11 oa1=1
12 nop
13 cjne ia1,1,12
14 oa1=0
.....
Stepping through the sequence with NEXT from step 13 to step 14 is only possible when user input A is high.
This command will also start a selected sequence when it is in STOP mode.

• Stop a Sequence

If the sequence mode is RUN or PAUSE , it can be stopped by the command:
Syntax : **PROGram:SELected:STAte<sp><STOP><term>**
The sequence will stop immediately, but is still selected.

• Read Sequence mode

By sending the query:
Syntax : **PROGram:SELected:STAte?<term>**
The SM15K will return the current mode. There are three possibilities:
STOP<term>
PAUSE,<next step><term>
RUN,<next step><term>
Syntax : **PROGram:SELected:STAte<SP>active?<term>**
The SM15K will return the current mode.
There are three possibilities:
STOP<term>
PAUSE,<active step><term>
RUN,<active step><term>

• Trigger a Step

When a step contains TRG, the sequence waits for the command via TCP/IP:
Syntax : **TRIGger:IMMediate<term>**
After this command, the sequencer will proceed.

• Add labels

Use the following command to define labels:
Syntax : **PROGram:SELected:LABel<sp><name>,<step><term>** name = Label name
To query the active Labels, use the command:
Syntax : **PROGram:SELected:LABel<sp>?<term>**

A maximum of 20 Labels can be defined.The maximum characters per Labelname is 10. Start with a A-Z character, after which A-Z, 0-9 characters are allowed.
See paragraph 6.6 for example.

• Delete labels

Delete a Label by sending the command:
Syntax : **PROGram:SELected:LABel<sp><NAMe>,DELETE<term>**
Use the following command to delete all Labels:
Syntax : **PROGram:SELected:LABel<sp><*>,DELETE<term>**

• Building a Sequence

Before a sequence can be started it has to be build. During a build the SM15K will, for example, check if all used Labels are defined properly.

If a sequence is edited, it can be build by the command:
Syntax : **PROGram:SELected:BUIld<term>**

To check if a selected sequence is build use the command:
Syntax : **PROGram:SELected:BUIld?<term>**

Note: executing a sequence by RUN or NEXT command without building it will automatically execute the Build command.

• Select saving to non-volatile memory

Initially a created and edited sequence is kept in volatile memory. This means that the sequence is lost when the Power Supply is switched off.

When a sequence is meant to remain on the Power Supply it can be set to be saved to non-volatile memory. Use the command:
Syntax : **PROGram:SELected:NONvolatile<sp><boolean><term>** <boolean> = 0 or 1, OFF,ON

The setting can be queried by the command:
Syntax : **PROGram:SELected:NONvolatile?<term>**

• Saving to non-volatile memory

Sequence set to be saved to non-volatile memory can be saved using the command:
Syntax : **PROGram:SAVe<term>**
A save takes approximately 5 seconds.
The current sequence save state can be queried by the command:

Syntax : **PROGram:SAVe?<term>**
The SM15K will return the current state. There are three possibilities:

0<term>	Sequence not saved yet.
1<term>	Sequence is being saved
2<term>	Sequence is saved

• Selecting a Programming Source

The programing source which is used by the sequencer can be selected. This gives the opportunity to for example control the unit with the front knobs and run a sequence which adjusts these knob settings. For example reducing Vset by 2V when a certain digital I/O is true.

Select the Programming Source using the command:
Syntax : **PROGram:SOUrce<sp><volt>,<curr>,<power><term>**

Possible settings are :
- Null
- Front
- Web
- Sequencer
- Ethernet
- Slot1 - Slot4 (except INT MOD ANA)

To Query the Programming Source set use the command:
Syntax : **PROGram:SOUrce?<term>**

6.4 Sequence control by Web

• Read the Catalog

The catalog contains all the programmed sequences. To read the catalog browse to the IP address of the power supply and click on Configuration > Sequences.

Sequencer Catalog

• How to create a sequence

When Sequencer control by Web is used, sequences have to be made in a text editor like for example Microsoft® Notepad and saved with a *.seq extension

Warning! do not save with a *.seq.txt extension.

The file name (excluding the start condition and the extension) defines the sequence name. It may contain the characters A-Z, 0-9 and + (max. 16 characters). The first character of the filename has to be an A-Z, type character. Sequence names must be unique, but are not case sensitive.

Build the actual sequence with step numbers, sequence commands and labels (one command per line). Step numbers must be in ascending order, but may be left out for future use. The power supply will skip unused sequence steps. See the example below.

Leave a space or a tab between the step number and the sequence command. Using a tab gives the opportunity to prepare the sequence in a spreadsheet program before copying it to a text editor.

A Linefeed (Enter) is needed after each sequence step. The power supply uses this linefeed to distinguish each command line. Make sure to add a linefeed after the last sequence step too.

Sequence example code: (Note. The unused steps 6 – 19 are skipped by the power supply)

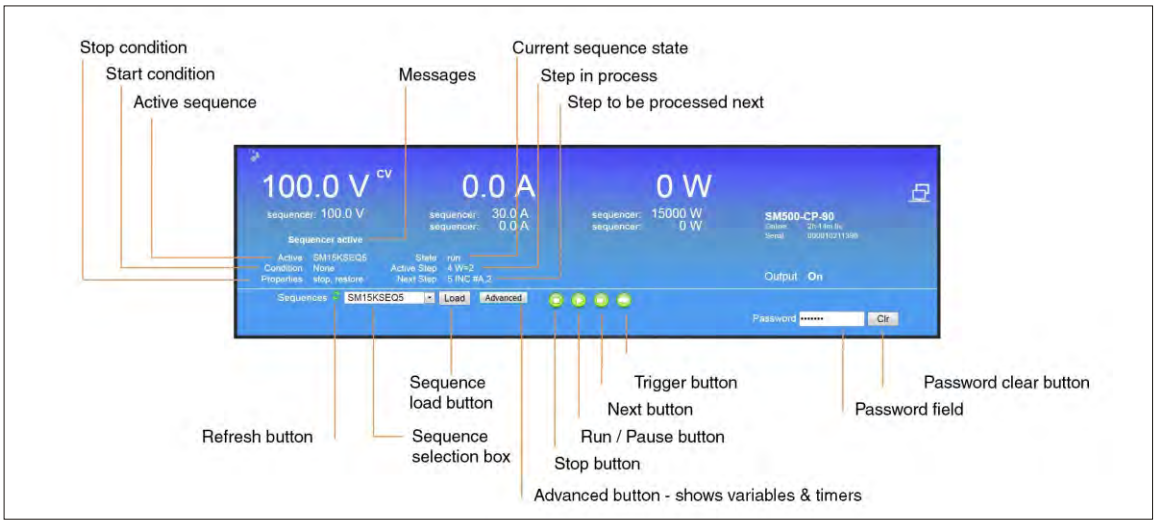
```
1 sc=8
2 sv=100
3 sp=15000
4 #j=3
5 inc sv,5
20 cjne #j,0,5
21 sv=100
…
```

• How to select a sequence

Sequences available in the power supply can be selected using the Sequencer Console. Browse to the IP address of the power supply and click on Console > Sequencer. The Console will display a selection box with which a sequence can be selected. Select the required sequence and click on Load. The sequence is now selected.

Note. If the selection box does not show the newly added sequence the Refresh button on the left side of the selection can be clicked.

The unfold menu of a sequence in the sequence catalog also mentions whether a sequence is active or not.



Sequencer console

• Upload a sequence to SM15K (PC → PS)

Click on Browse on the catalog page to browse to the sequence file. Select the file, click on Open and click on upload to start uploading the file. When the sequence is uploaded the power supply checks its Syntax and performs the Build command (section 6.3 Building a Sequence) Click on Back to find the sequence in the catalog when the upload is accepted by the power supply.

• Download a Sequence from SM15K (PS → PC)

Unfold the sequence options by clicking on the sequence name in the catalog. Click on <sequence name>.seq next to Download source to download the sequence and select a text editor to open the file.

• Delete a Sequence

Unfold the sequence options by clicking on the sequence name in the catalog. Click on the Selection box next to Delete and click on the button Apply settings. Use a password when required. Make sure that the sequence is stopped before clicking on Apply settings, because a running or paused sequence cannot be deleted. The Sequencer Console can be used to stop a running or paused sequence.

When the sequence has been saved to non-volatile memory (the checkbox Mark for Non-volatile will be checked) the checkbox Mark for Non-volatile needs to be unchecked and applied as well. Additionally click on Sync memory to update the non-volatile memory. Updating the non-volatile memory takes about 5 seconds in which the web will wait.

• Start a Sequence

A selected sequence can be started by the Run/Pause button on the Sequencer Console. Unless Paused, the sequence will start from step 1.

• Pause a Sequence

A running sequence can be paused by clicking on the Run/Pause button on the Sequencer Console. Line <next step> will visualize the next command to be executed.

Step through a Sequence

If a sequence is selected, it is possible to manually step through the program (during any mode). Click on the Next button on the Sequencer Console to execute the next step.

Line <next step> will visualize the next command to be executed.

• Stop a Sequence

If the sequence mode is Run or Pause, it can be stopped by the Stop button on the Sequencer Console

• Read Sequence mode

State <current state> on the Sequencer Console displays the current state. There are three possibilities: Stop, Pause and Run. The Pause and Run states might additionally mention trigger when the sequence waits for a human, or software trigger interaction.

• Trigger a Step

When a sequence contains TRG, the sequence waits for the trigger button. This is equal to the TRIG:IMM command via TCP/IP (section 6.3 Trigger a step).

• Add labels

Labels can be defined and used in a sequence file. During sequence upload the power supply checks if all used labels are also defined in the file.

To define a label, type its name and add a : (colon) character to it. The sequencer will jump to the sequence step directly below the label define. Use the label name in sequence steps without the colon character.

Label example code:

```
...
234 w=2
increase:
235 inc sv,0.5
236 w=0.2
237 cjl sv,14,increase
238 w=10
…
```

A maximum of 20 labels can be defined per sequence. The maximum characters per label name is 10. Start with a A-Z character, after which A-Z, 0-9 type characters are allowed.

• Delete labels

To delete a label, remove its define and the relevant jumps from the sequence file on the computer. It is not possible to edit uploaded sequences using the browser.

• Building a Sequence

Directly after sequence file upload the power supply automatically builds the sequence to usable core code. No more manual action is required. Unfolding the sequence options displays the Built state of a sequence. This can be No when the sequence is created using the commands mentioned in section 6.3 (Sequence control by commands).

• Rebuilding a Sequence

Rebuilding of a sequence is required when the system has changed. For example when the number of units in a M/S system has changed. The sequencer keeps track of this and will display a warning in such cases (on the web page, under configuration sequencer).

• Saving a sequence to non-volatile memory

Unfold the sequence options by clicking on the sequence name in the catalog. Click on the Selection box next to Mark for Non-volatile and click on the button Apply settings. Use the password when required. Click on Sync memory to update the non-volatile memory with the new setting. Updating the non-volatile memory takes about 5 seconds in which the web will wait.

• Selecting a Programming Source

The programming source which is used by the sequencer can be selected. Refer to 6.3 (Sequence control by commands) for the Ethernet command. Selecting the Sequencer programming source by Web is not implemented.

• Sequence control by user inputs (optional)

Unfolding the sequence options displays Start conditions. When the power supply is equipped with one or multiple Digital I/O interfaces it is possible to start and stop a sequence on a user input. Refer to section 6.5 (Sequence control by user inputs) for the description on the sequence control by user input settings.

6.5 Sequence control by user inputs (optional)

When a new sequence is created (by prog:sel:name), an assignment can be added to the sequence name to give the sequence extra parameters.
Once sequences are uploaded, it is possible to start / stop them by user inputs. This gives the opportunity to control the sequencer without the need of a computer. Even the network connection can be removed. To do so it is necessary to insert 1 or multiple Digital I/O Interfaces (see section 5.6.1). The power supply will be able to work stand-alone in the field and control up to 25[1] different complex processes (one at a time).

There are some rules that must be followed:
1) To start a sequence, the related user input must change from "0" to "1".
2) To stop / finish a sequence, the related user input must change from "1" to "0".
3) To start another sequence, the other assigned user inputs must be "0".

The assignment of a user input to a sequence must be included within the sequence name.
A sequence name can be max 16 characters long. The last four characters can be used for assignment. First the separator "**+**", followed by **A,B,...,G or H** (the name of a user input) and the slot number.

Character 3 and 4 allow to set some options:

3th:	**S** (Stop)	;	When the assigned user input changes from "1" to "0", the sequence will stop immediately.
	F (Finish)	;	When the assigned user input changes from "1" to "0", the sequence continues until the command END is executed.
4th:	**R** (Restore);		When the sequence stops, the SM15K restores the voltage, current and power settings that where valid before the sequence started.
	H (Hold)	;	When the sequence stops, the actual voltage, current and power settings remain the same.

Assignment examples:

<seq name>**+A1SR** ; The sequence starts when user input A (slot 1) becomes "1", it stops immediately when user input A (slot 1)becomes "0" and the voltage, current and power settings are restored.

<seq name>**+D3FH** : The sequence starts when user input D (slot 3) becomes "1". When D (Input slot 3)becomes "0", it finishes the steps until END is executed and holds the actual setting for voltage, current and power.

Any combination of the user inputs and the options can be made. It is required to set every character of the assignment. Refer to section 6.3 to find the command to program the sequence name. The length of the sequence name + the assignment has a maximum of 16 characters.

1) When 4 Digital I/O Interfaces are inserted

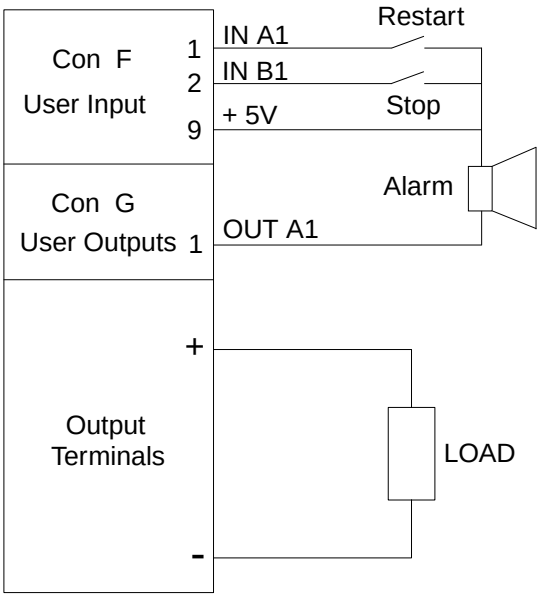
6.6 Sequence examples

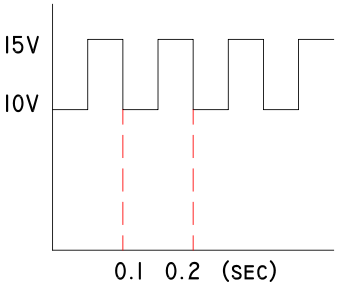
In this paragraph a few examples are explained based on typical applications.
Every example contains a short description of the application, and the list of the required sequence steps.

Example 1: Generate waveform.

A rectangular waveform (10Hz) with an amplitude of 5V and an offset of 10V must be generated by the power supply. If the output current of the power supply becomes below 26 Amperes, the output voltage must drop to 0V and an alarm bell must indicate the fault. One push button restarts the system, and another one stops the system.

Wiring diagram:





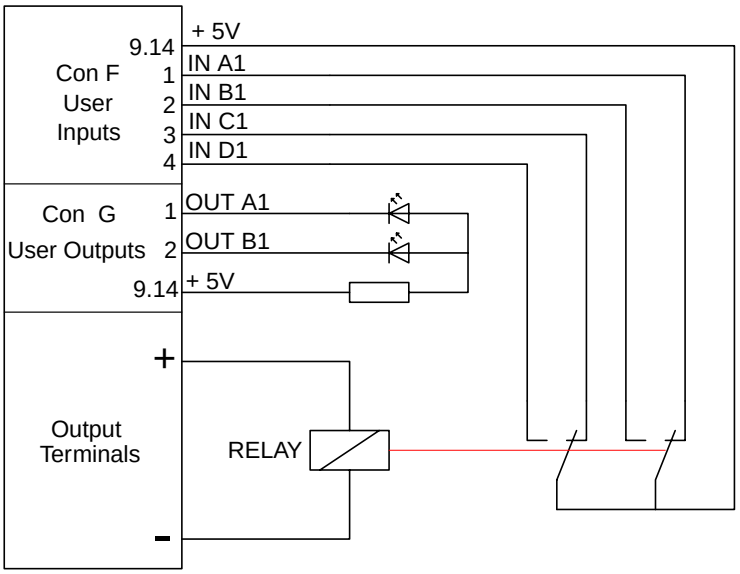
Programming steps:

```
1 sv=0
2 sc=45
3 sp=15000
begin:
4 oa1=0
5 w=1
repeat:
6 sv=10
7 w=0.05
8 sv=15
9 w=0.05
10 cje ib1,1,stop
11 cjg mc,26,repeat
12 sc=0
13 sv=0
14 oa1=1
restart:
15 cjne ia1,1,restart
16 jp begin
stop:
17 sv=0
18 sc=0
19 end
```

Example 2: Test relays.

To test the double-pole contacts of relays and the working voltage of their coils (specified between 6 and 11.8V), the output voltage of the power supply ramps from 5 to 12 Volt. At 5V, the output current must be higher than 10mA, otherwise the test doesn't start.
Contacts are tested open and closed. When the relay switches, the working voltage must be checked. The results are indicated via a red or a green LED.

Wiring diagram:



Programming steps:

```
1 oa1=0
2 ob1=0
3 js 20
4 nop
5 w=1
6 sv=5.9
7 cjne ia1,1,30
8 cjne ib1,0,30
9 cjne ic1,1,30
10 cjne id1,0,30
11 cjg sv,11.8,30
12 inc sv,0.05
13 w=0.1
14 cjne ia1,1,34
15 cjne ib1,0,34
16 cjne ic1,1,34
17 cjne id1,0,34
18 jp 11
19 end
20 sv=5
21 sc=0.3
22 sp=25
23 w=0.1
24 cjg mc,0.01,29
25 oa1=1
26 ob1=1
27 w=1
28 jp 19
29 ret
30 oa1=1
31 w=1
32 jp 19
33 nop
34 ob1=1
35 w=1
36 jp 19
37 nop
```