

PSC-ETH

General information

Installation manual

Command description

Table of Contents:

1	In General	page 1 - 1
1.1	Features	page 1 - 1
1.2	Analog	page 1 - 1
1.3	Status	page 1 - 1
1.4	LEDs	page 1 - 1
1.5	Digital I/O	page 1 - 2
2	Installation	page 2 - 1
2.1	Infra structure	page 2 - 1
2.2	Settings	page 2 - 1
2.3	Terminal	page 2 - 1
3	Communication	page 3 - 1
3.1	Settings	page 3 - 1
3.2	TCP/IP	page 3 - 1
3.3	LabVIEW™ driver	page 3 - 1
3.4	UDP	page 3 - 1
4	Conventions	page 4 - 1
4.1	Syntax	page 4 - 1
4.2	Query	page 4 - 1
4.3	Space	page 4 - 1
4.4	Line feed	page 4 - 1
4.5	Parameters	page 4 - 1
5	Command description	page 5 - 1
5.1	General Instructions	page 5 - 1
	*IDN?	page 5 - 1
	*PUD	page 5 - 1
	*SAV	page 5 - 1
	*RST	page 5 - 2
	*RCL	page 5 - 2
5.2	Source Subsystem	page 5 - 3
	Introduction	page 5 - 3
	Maximum Output Voltage	page 5 - 3
	Maximum Output Current	page 5 - 3
	Set Output Voltage	page 5 - 3
	Set Output Current	page 5 - 3
5.3	Measure Subsystem	page 5 - 4
	Introduction	page 5 - 4
	Measure Output Voltage	page 5 - 4
	Measure Output Current	page 5 - 4
	Measure Output Power	page 5 - 4
5.4	Calibrate Subsystem	page 5 - 5
	Introduction	page 5 - 5
	Calibrate Gain Source Current	page 5 - 6
	Calibrate Offset Source Current	page 5 - 6
	Calibrate Gain Source Voltage	page 5 - 6
	Calibrate Offset Source Voltage	page 5 - 6
	Calibrate Gain Measure Current	page 5 - 6
	Calibrate Gain Measure Voltage	page 5 - 6
	Calibrate Offset Measure Current	page 5 - 6
	Calibrate Offset Measure Voltage	page 5 - 6
5.5	Digital User In-/Outputs	page 5 - 7
	User Outputs	page 5 - 7
	User Inputs	page 5 - 7

5.6	System Subsystem	page 5 - 8
	Remote Shut Down (RSD)	page 5 - 8
	Front Panel Lock	page 5 - 8
	Remote programming	page 5 - 8
	Error Message	page 5 - 9
	Password	page 5 - 9
5.7	Output	page 5 - 10
5.8	Register Structure	page 5 - 11
	Introduction	page 5 - 11
	Condition Register	page 5 - 11
	Positive Transition Register (PTR)	page 5 - 11
	Negative Transition Register (NTR)	page 5 - 11
	Event Register	page 5 - 12
	Enable Register	page 5 - 12
	*ESR?	page 5 - 13
	*ESE	page 5 - 13
	*ESE?	page 5 - 13
	*STB?	page 5 - 13
	*SRE	page 5 - 13
	*SRE?	page 5 - 14
	*CLS	page 5 - 14
	User Input Registers	page 5 - 14
	Condition	page 5 - 14
	PTR	page 5 - 14
	NTR	page 5 - 14
	Event	page 5 - 14
	Enable	page 5 - 14
5.9	Status Operation Register Set	page 5 - 15
	Status Operation Registers	page 5 - 15
	Condition	page 5 - 15
	PTR	page 5 - 15
	NTR	page 5 - 16
	Event	page 5 - 16
	Enable	page 5 - 16
	Status Operation Settings Registers	page 5 - 16
	Conffdition	page 5 - 16
	PTR	page 5 - 16
	NTR	page 5 - 16
	Event	page 5 - 16
	Enable	page 5 - 16
	Status Operation Regulating Registers	page 5 - 16
	Condition	page 5 - 16
	PTR	page 5 - 16
	NTR	page 5 - 16
	Event	page 5 - 16
	Enable	page 5 - 17
	Status Operation RControl Registers	page 5 - 17
	Condition	page 5 - 17
	PTR	page 5 - 17
	NTR	page 5 - 17
	Event	page 5 - 17
	Enable	page 5 - 17
	Status Operation Shutdown Registers	page 5 - 17
	Condition	page 5 - 17
	PTR	page 5 - 17
	NTR	page 5 - 17
	Event	page 5 - 17
	Enable	page 5 - 17

	Status Operation Shutdown Protection Registers	page 5 - 17
	Condition	page 5 - 17
	PTR.....	page 5 - 18
	NTR	page 5 - 18
	Event.....	page 5 - 18
	Enable	page 5 - 18
5.10	Status Questionable Register Set	page 5 - 19
	Status Questionable Registers	page 5 - 19
	Condition	page 5 - 19
	PTR.....	page 5 - 19
	NTR	page 5 - 19
	Event.....	page 5 - 19
	Enable	page 5 - 19
	Status Questionable Voltage Registers.....	page 5 - 20
	Condition	page 5 - 20
	PTR.....	page 5 - 20
	NTR	page 5 - 20
	Event.....	page 5 - 20
	Enable	page 5 - 20
	Status Questionable Current Registers.....	page 5 - 20
	Condition	page 5 - 20
	PTR.....	page 5 - 20
	NTR	page 5 - 20
	Event.....	page 5 - 20
	Enable	page 5 - 20
6	Sequencer.....	page 6 - 1
6.1	Introduction	page 6 - 1
6.2	Commands.....	page 6 - 1
	Settings.....	page 6 - 1
	Jumps	page 6 - 2
	Arithmetic.....	page 6 - 3
	Miscellaneous	page 6 - 3
6.3	Sequence control by commands	page 6 - 4
6.4	Sequence control by user inputs	page 6 - 5
6.5	Sequence examples	page 6 - 7
	Example 1: Generate waveform	page 6 - 7
	Example 2: Test relays.....	page 6 - 8
7	Command list TCP/IP	page 7 - 1
8	Command list Sequencer.....	page 8 - 1

1 In General

1.1 Features

The PSC-ETH is an interface between a TCP/IP network and a Power Supply. It allows the operator to create fully automated systems. The PSC-ETH can set and measure the parameters of the power supply, reads the status, sets controls, interacts with digital I/O and generates 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 PSC-ETH is designed especially for these kinds of power supplies. Therefore the programming and measuring resolution is 16 bits and all units are calibrated very precisely by the factory (if unit and PSC are bought together). To calculate the step size in (Voltage or Ampere) use the equation below:

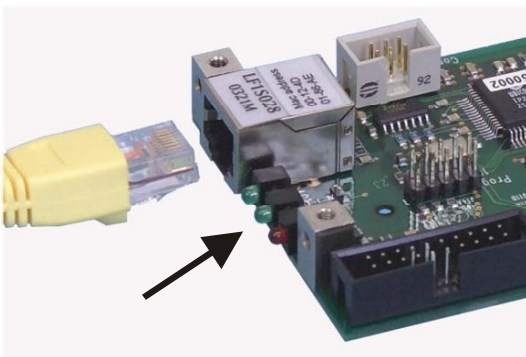
$$\text{StepSize} = \frac{\text{MaximumOutput}}{2^{16}}$$

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. It is also possible to let the PSC-ETH generate a so called "Service Request"-message (SRQ) as soon as an assigned status has changed, without the need of continuously reading the status itself. This makes the internal register structure act as a sort of a watchdog. Refer to section 5.8 to learn more about this feature.

1.4 LEDs

Near the RJ45 connector of the PSC-ETH there are three LEDs, see picture below.



The red LED is named "ERR". When this LED is ON, the PSC-ETH has generated an error message as a result of a bad command or parameter.

The middle green LED is named "LNK" (Link) and is ON when the PSC-ETH is connected to the network.

The left green LED is named "ACT", which stands for Activity. This LED indicates whether or not the PSC-ETH communicates.

1.5 Digital I/O

The SM1500, SM6000 and the external PSC-ETH provide 8 user inputs and 6 outputs. These can be controlled / monitored by commands (refer to section 5.5) or can be used to interact with the Sequencer (refer to chapter 6) to make the power supply act like a power PLC.

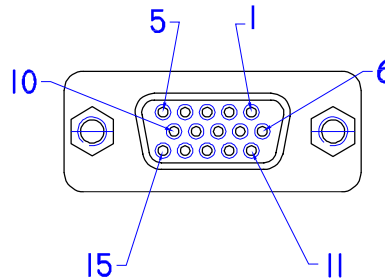
User Inputs (connector CON F):

Unconnected inputs are LOW. To make an input HIGH, apply the +5V of pin 9 or pin 14. Also an external voltage can be used to control the inputs. Make sure the common of the external source is connected to pin 10 or pin 15.

The user inputs have a working range of 2.5V - 30V and have a internal impedance of approximately 2KOhms. The maximum load of the +5V is 100mA.

Con F (female)

- 1 = user input A
- 2 = user input B
- 3 = user input C
- 4 = user input D
- 5 = user input E
- 6 = user input F
- 7 = user input G
- 8 = user input H
- 9 = +5V
- 10 = 0V
- 11 = n.c.
- 12 = n.c.
- 13 = n.c.
- 14 = +5V
- 15 = 0V



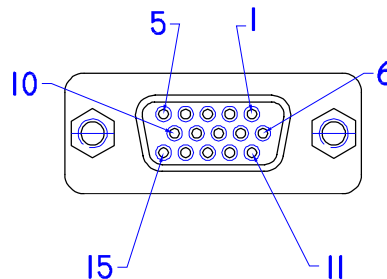
User Outputs (connector CON G):

The 6 user outputs are open collector. When an output is TRUE, it is pulled down to 0V.

Otherwise the output is open. Check the datasheet for maximum voltage and sink current of the user outputs.

CON G (female)

- 1 = user output A
- 2 = user output B
- 3 = user output C
- 4 = user output D
- 5 = user output E
- 6 = user output F
- 7 = do not use
- 8 = do not use
- 9 = +5V
- 10 = 0V
- 11 = n.c.
- 12 = n.c.
- 13 = n.c.
- 14 = +5V
- 15 = 0V



2 Installation

2.1 Infra structure

To control or program the PSC-ETH, it should be connected to a TCP/IP network, using the RJ45 connector at the rear (connector CON H). A standard RJ45 patch cable can be used to connect the PSC-ETH to a network hub or switch. To connect the PSC-ETH directly to the network connection of a computer an RJ45 cross-link cable should be used. The "LNK"- LED of the PSC-ETH will be on when the connection is okay.



The command "PING <address>" of Windows can be used (via menu Start - RUN - CMD) to check the connection. For example : PING 10.1.101

Use the program "Configurator" to find out the actual address of the PSC-ETH, see chapter 2.2.

2.2 Settings

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

The setup installs a program named "Configurator". This program shows the IP addresses of the PSC-ETHs connected to the network and allows to change it. It also shows the MAC address, the serial number and the identification string (*PUD) of every PSC-ETH.

By clicking on (one of) the PSC-ETH(s), the user can set the IP address, the default gateway and the IP address mask.

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 : 255.255.0.0)
Class C :	192.168.0.0	to	192.168.255.255	(mask : 255.255.255.0)

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 PSC-ETH will be out of range. That is why this program blocks these address settings.

To set the IP address back to default, click on "file, set default". Type the MAC-address (which can be found on a label on the PSC-ETH card) and press <Enter>. The IP address will then be 10.1.0.101.

(The program "IPCONFIG" of Windows can be used (via menu Start - RUN - CMD) to check the address and mask of the local host.)

When more PSC-ETH devices are used, it is recommended to give them specific names, using the command *PUD (refer to chapter 5.1).

2.3 Terminal

The setup also installs the program named "PSC-ETH Terminal". This program can be used to send commands to and read queries from the PSC-ETH easily. Very useful for checking the connection or to send commands manually.

After installation, this program can be found under "Programs \ Delta Elektronika". First set the right IP address in the "IP address"-box. Then type a command (e.g. *IDN?) in the "Characters to Write"-box and click on the "Send"-button.

The commands to be sent are checked for a question mark. If present, the terminal waits for the answer from the PSC-ETH, identified with an "auto query" LED on the screen after a command is sent. The answer appears in the "Characters Read"-box.

3 Communication

3.1 Settings

There are two important settings to communicate properly with the PSC-ETH, which are:
IP address : default 10.1.0.101. Can be freely configured by the user (refer to section 2.2).
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 PSC-ETH. To show the user / programmer how to communicate with the PSC-ETH, examples of Visual Basic and LabVIEW are provided on the CD_ROM.

3.3 LabVIEW™ driver

LabVIEW™ is a graphical programming language of National Instruments. This language is very powerful and easy to use. It allows the programmer to build applications within a short time. Delta Elektronika BV created a LabVIEW™ driver to control the PSC-ETH without the need for the programmer to know the details of communication and commands. A driver for LabVIEW™ 7 is provided. Check our website for updates : www.deltapowersupplies.com

For communication between LabVIEW™ and the PSC-ETH a DLL (Dynamic Link Library) is created. The LabVIEW™ driver uses this DLL to open and close the connection and to write and read data. A big advantage of this DLL is, that a command is sent in a very small TCP/IP packet (one command per package). Compared to the standard TCP/IP functions, where large packages are sent with the change of more than one command per package.

3.4 UDP

Once a connection is made, the PSC-ETH is capable of sending messages via UDP (User Datagram Protocol), via port number 8462. These messages can be used to inform the connected host / user about changes of predefined status.

The PSC-ETH has a built-in register structure which allows the user to define which status (one or more) will generate a message when it changes.

For example the user can be prompted by a UDP message when the power supply has an AC failure (missing or incorrect input voltage) or turns in CC-mode.

The UDP message has two parts: first two characters indicate the type of the message, the last two characters show the value of the *STB register. (All characters are in HEX format).

Type of message :

- 01 : Service ReQuest (SRQ)
- > 02 : reserved for future implementation.

Refer to section 5.8 to learn more about the registers and the SRQ.
The CD-ROM contains an example how to use this feature within LabVIEW7.

4 Conventions

The PSC-ETH 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 PSC-ETH.

It is allowed either to use the short form (uppercase) or the entire command.

For example : SOURce:VOLTage<sp>5<nl> can be send as :

sour:volt<sp>5<nl>

source:volt<sp>5<nl>

source:voltage<sp>5<nl>

sour:voltage<sp>5<nl>

SoUrCe:VoLt<sp>5<nl>

(Sending mixed upper- and lowercase is allowed).

4.2 Query

If a command ends with a question mark "?", a query is sent to the PSC-ETH.

4.3 Space

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

4.4 Line feed

At the end of a command a line feed will be sent. Also every answer coming from the PSC-ETH is terminated by a line feed, <nl>. This is equal to ASCII character 0AH (10d).

4.5 Parameters

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

<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 PSC-ETH. The string contains the name, the option number, the version of the firmware and the serial number of the PSC.

Syntax : ***IDN?<nl>**

The response string has 4 fields, separated by commas.

For example :

DELTA<sp>ELEKTRONIKA<sp>BV,PSC<sp>ETH<sp>P157<sp>V1.0.0,449101000099,0<nl>

The first field shows the manufacturer's name.

The second field shows : interface name (PSC ETH) + option name (P157) + firmware version.

The third field shows the serial number of the PSC-ETH interface.

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 March 5 2005".

This information can be maximum 72 characters long and is stored into non-volatile memory (see command *SAV).

Syntax : ***PUD<sp><data><nl>** data = any characters, 72 maximum

To read the Protected User Data:

Syntax : ***PUD?<nl>**

*SAV

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

A summary of the saved setting are shown below:

Maximum output voltage	: SOURce:VOLTage:MAXimum
Maximum output current	: SOURce:CURRent:MAXimum
Calibration gain current setting	: CAL 0
Calibration offset current setting	: CAL 1
Calibration gain voltage setting	: CAL 2
Calibration offset voltage setting	: CAL 3
Calibration gain current measure	: CAL 4
Calibration gain voltage measure	: CAL 5
Calibration offset current measure	: CAL 6
Calibration offset voltage measure	: CAL 7
Protected User Data	: *PUD
Password	: SYSTem:PASSword

Syntax :

***SAV<nl>**

When a password is used, the only way to store this relevant settings into memory is to use:

Syntax : ***SAV<sp><password><nl>**

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

***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 PSC-ETH:

Setting	Value	set after *RST:	set after power-on:
SOURce:VOLTage	0	YES	YES
SOURce:CURREnt	0	YES	YES
SYSTem:RSD	OFF	YES	YES
SYSTem:REM	REM	YES	NO
SYSTem:REM:CV	REM	YES	NO
SYSTem:REM:CC	REM	YES	NO
SYSTem:FRONtpanel	OFF	YES	NO
OUTPut	OFF	YES	NO

***RCL**

This command Recalls CaLibration settings and the Protected User Data. This can be used in case of accidental override of these settings.

Syntax : ***RCL<nl>**

5.2 Source Subsystem

Introduction

First of all the PSC-ETH should know which type of power supply it will be controlling. Therefore two commands are available to set the maximum output voltage and the maximum output current. Default settings are 5V / 5A. These settings must be changed for correct operation. When the PSC-ETH is built-in by Delta Elektronika BV, these settings are already done.

Both parameters (output voltage and output current) have a working range from 0 to the maximum output voltage / current of the power supply.

The response strings contain a floating point value with 4 digits of precision.

Maximum Output Voltage

To set the maximum output voltage of the power supply, this command must be sent. Refer to section 5.1 how to save this setting to the non-volatile memory.

Syntax : **SOURce:VOLTage:MAXimum<sp><NR2><nI>**

e.g. <NR2> is 30 for an SM30-200 power supply.

To read the last settings, send the query:

Syntax : **SOURce:VOLTage:MAXimum?<nI>**

For example the answer is: 30.0000<nI>

Maximum Output Current

To set the maximum output current of the power supply, this command must be sent. Refer to section 5.1 how to save this setting to the non-volatile memory.

Syntax : **SOURce:CURRent:MAXimum<sp><NR2><nI>**

where e.g. <NR2> is 200 for an SM30-200 power supply.

To read the last settings, send the query:

Syntax : **SOURce:CURRent:MAXimum?<nI>**

Set Output Voltage

To set the output voltage of the power supply:

Syntax : **SOURce:VOLTage<sp><NR2><nI>**

To read the last settings, send the query:

Syntax : **SOURce:VOLTage?<nI>**

Set Output Current

To set the output current of the power supply:

Syntax : **SOURce:CURRent<sp><NR2><nI>**

To read the last settings, send the query:

Syntax : **SOURce:CURRent?<nI>**

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 SOURCE:VOLTage and SOURRe:CURRent.

If for example the output settings are 15V and 5A and the unit is in CC-mode (Constant Current), the output voltage is not equal to the setting of 15V, but less.

The resolution of the answers are 16 bits, displayed with 4 digits of precision.

Measure Output Voltage

To measure the output voltage of the power supply:

Syntax : **MEASure:VOLTage?<nl>**

Measure Output Current

To measure the output current of the power supply:

Syntax : **MEASure:CURRent?<nl>**

Measure Output Power

To measure the output power of the power supply, send this query. (Actually this command executes MEASure:VOLTage and MEASure:CURRent, multiplies the results and returns the output power in Watts)

Syntax : **MEASure:POWer?<nl>**

5.4 Calibrate Subsystem

Introduction

To make the accuracy of the PSC-ETH / power supply combination as high as possible, it is required to calibrate the PSC-ETH first. There are eight parameters to calibrate. Four related to the voltage settings / measurements and four related to the current settings / measurements.

The calibration settings are already done, when the PSC-ETH is built-in by Delta Elektronika BV. However, periodical check and calibration is recommended. At power-on and after a reset, the calibration settings are restored.

For proper calibration it is recommended to use this order (an SM60-100 is used as example) :

- Set output voltage of the power supply to e.g. 1% of the maximum output voltage.
SOUR:VOLT<sp>0.6<nI>
- Calibrate the source voltage offset, so the output voltage is as close as possible to 0.6V
- Calibrate the measure voltage offset, so the result of the command MEASure:VOLTage? is as close as possible to 0.6V.
- Set the output voltage to maximum
- Calibrate the source voltage gain, so the output voltage is as close as possible to the maximum voltage, 60V.
- Calibrate the measure voltage gain, so the result of the command MEASure:VOLTage? is as close as possible to the actual output voltage, 60V.

Same principle is used for the current calibration.

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

For SOURCE offset calibration, use the equation:

$$new_offset = old_offset + \left(\frac{programmedValue - actualValue}{MaximumOutput} \right) * 5$$

For SOURCE gain calibration, use the equation:

$$new_gain = old_gain * \left(\frac{programmedValue}{actualValue} \right)$$

For MEASURE offset calibration, use the equation:

$$new_offset = old_offset + \left(\frac{actualValue - measuredValue}{MaximumOutput} \right) * 5$$

For MEASURE gain calibration, use the equation:

$$new_gain = old_gain * \left(\frac{actualValue}{measuredValue} \right)$$

settings and readings are 6 digits.

Be aware : both gain and offset changes has influence to 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.

Calibrate Gain Source Current

To calibrate the programming gain of the current setting:

Syntax : **CAL**<sp>**0**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**0?**<nl>

Calibrate Offset Source Current

To calibrate the programming offset of the current setting:

Syntax : **CAL**<sp>**1**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**1?**<nl>

Calibrate Gain Source Voltage

To calibrate the programming gain of the voltage setting:

Syntax : **CAL**<sp>**2**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**2?**<nl>

Calibrate Offset Source Voltage

To calibrate the programming offset of the voltage setting:

Syntax : **CAL**<sp>**3**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**3?**<nl>

Calibrate Gain Measure Current

To calibrate the gain of the current measurement:

Syntax : **CAL**<sp>**4**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**4?**<nl>

Calibrate Gain Measure Voltage

To calibrate the gain of the voltage measurement:

Syntax : **CAL**<sp>**5**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**5?**<nl>

Calibrate Offset Measure Current

To calibrate the offset of the current measurement:

Syntax : **CAL**<sp>**6**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**6?**<nl>

Calibrate Offset Measure Voltage

To calibrate the offset of the voltage measurement:

Syntax : **CAL**<sp>**7**,<NR2><nl>

To read the calibration settings:

Syntax : **CAL**<sp>**7?**<nl>

5.5 Digital User In-/Outputs

The SM1500, SM6000 and the external PSC-ETH provide 8 user inputs and 6 outputs. These can be controlled / monitored by commands (explained below) or can be used to interact with the Sequencer (refer to chapter 6).

User Outputs

To program the user outputs, a decimal number can be send to the PSC-ETH. This decimal number represents the binary state of the 6 user outputs.

Syntax : **UOUTput<sp><0+NR1><nl>** 0+NR1= 0,1,2,3.....63

Output A = 1
 Output B = 2
 Output C = 4
 Output D = 8
 Output E = 16
 Output F = 32

For example, to set output C and F, send **UOUTput<sp><36><nl>**. (=4+32)

To reset all the user outputs, send **UOUTput<sp><0><nl>** (default setting after power-on).

To read the last setting of the user outputs:

Syntax : **UOUTput?<nl>**

User Inputs

To read the status of the 8 user inputs, the User Input Condition Register can be read:

Syntax : **UINPut:CONDition?<nl>**

The PSC-ETH 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<nl>**. (=1+64)

There is also a possibility to generate an SRQ message via UDP after a change on one of the user inputs. Therefor some settings in the register structure are required. Refer to section 5.8, 5.9, 5.10 to learn more about this feature.

5.6 System Subsystem

Remote Shut Down (RSD)

Syntax : **SYSTem:RSD[:STATus]<sp><boolean><nl>** boolean = 0, 1, OFF, ON

To read the last setting:

Syntax : **SYSTem:RSD[:STATus]?<nl>**

Front Panel Lock

This command is only available for the PSC-ETH built in a power supply with internal CPU; SM6000, SM1500.

Syntax : **SYSTem:FRONTpanel[:STATus]<sp><boolean><nl>** boolean = 0, 1, OFF, ON

To read the last setting:

Syntax : **SYSTem:FRONTpanel[:STATus]?<nl>**

Remote programming

The built-in PSC-ETH allows the user to switch to either LOCAL or REMOTE programming.

There are two different methods to select the programming source, depending on the type of power supply. Check table 5.1.

Remote method 1

With this method the two programming signals CV and CC are switched simultaneously.

Syntax : **SYSTem:REMOte[:STATus]<sp><setting><nl>** setting = REMote or LOCal

To read the last setting:

Syntax : **SYSTem:REMOte[:STATus]?<nl>**

Remote method 2

This method provides to switch CV or CC programming individually, so one of the two can be controlled via the PSC-ETH and the other one via the pot-meter on the front panel. Any combination is possible. Hence there are two commands to set the programming source:

To set the CV programming source:

Syntax : **SYSTem:REMOte:CV[:STATus]<sp><setting><nl>** setting = REMote or LOCal

To read the last setting:

Syntax : **SYSTem:REMOte:CV[:STATus]?<nl>**

To set the CC programming source:

Syntax : **SYSTem:REMOte:CC[:STATus]<sp><setting><nl>** setting = REMote or LOCal

To read the last setting:

Syntax : **SYSTem:REMOte:CC[:STATus]?<nl>**

Table of possible remote commands referred to the different power supplies:

TABEL 5.1	REMOTE	REMOTE:CV	REMOTE:CC
External	no	no	no
ES150	no	no	no
ES300	no	no	no
SM1500-1*	yes	no	no
SM1500	no	yes	yes
SM3000	no	no	no
SM6000	no	yes	yes

* SM1500-1 can be recognized by it's mains toggle switch ; instead of a rotary switch.

Error Message

If an unknown command, an invalid value or an illegal setting is received by the PSC-ETH an error is generated. The user is prompted by the red LED near the RJ45 connector and a 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 5 errors; more error messages are ignored. The command to read the queue line by line is:

Syntax : **SYSTem:ERRor?<nl>**

The PSC-ETH 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<nl>**

So after 5 readings of SYSTem:ERRor? the queue is empty for sure.

Password

To protect the most essential settings of the system (calibration values, maximum voltage / current, *PUD) a password can be used. As a default, no password is set.

To apply a password, send the command:

Syntax : **SYSTem:PASSword<sp><old_password>,<new_password><nl>**

If no password is used, <old_password> must be "**DEFAULT**" (upper case !) 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**" (upper case !).

Maximum length of password is 7 characters.

Be careful : There is no way to erase a password when it is unknown or forgotten.

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?<nl>**

If no password is used, the PSC-ETH will return 0<nl>, otherwise it returns 1<nl>.

5.7 Output

The PSC-ETH provides a command to switch the output of a power supply ON or OFF. This is only supported for power supplies, which have a push button "OUTPUT ON" on the front panel. Refer to the manual of the power supply to learn more about this function.

Syntax : **OUTPut**<sp><boolean><nl> boolean = 0, 1, OFF, ON

To read the last settings:

Syntax : **OUTPut?**<nl>

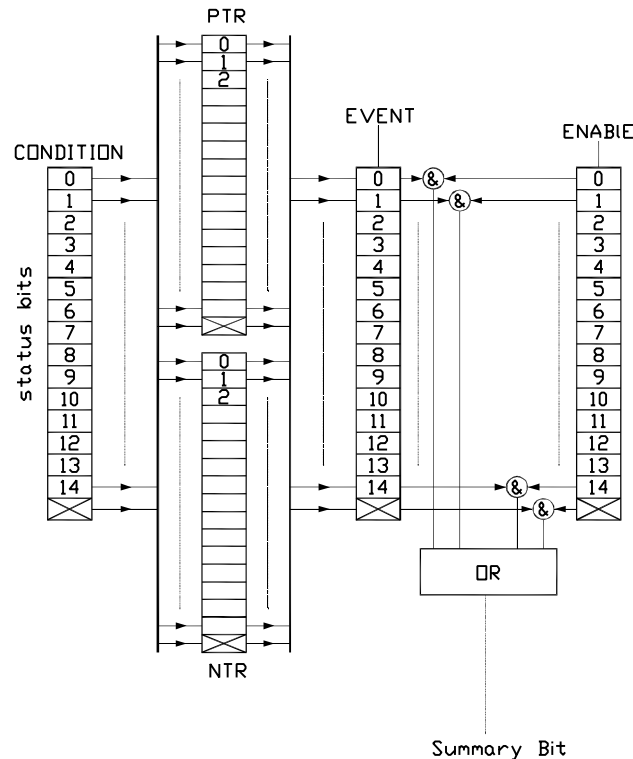
5.8 Register Structure

Introduction

The PSC-ETH has a complex structure of registers to be able to generate interrupts to the user in the case the status of the power supply changes. All registers can be set or cleared, so a user define interruption filter can be made.

The different status are grouped into Condition Registers (which can contain maximum 15 status). Each register has a PTR (Positive Transition Register), an NTR (Negative Transition Register), an Event Register and an Enable Register.

All registers are set and read by decimal numbers, which represent the binary state of the bits.



Condition Register

In the Condition Registers, the actual value of the status can be found. These registers are updated constantly.

Attention : the *actual* value can be different from the *last setting*. If for example the command is sent to disable the output (OUTPUT OFF), and after that the front panel's output button is pressed, the command OUTPUT? returns "0" (*last setting*), but the corresponding register (STATUS:OPERATION:SHUTDOWN:CONDITION) will tell you the output is ON (*actual value*).

Positive Transition Register (PTR)

By setting the bits in this register, the changes from "low" to "high" (0 to 1) of the bits in the corresponding condition register will set the event bits of the corresponding Event register.

After power-on the default values of all PTRs are "0", so no event will be generated.

Negative Transition Register (NTR)

By setting the bits in this register, the changes from "high" to "low" (1 to 0) of the bits in the corresponding condition register will set the event bits of the corresponding Event register.

After power-on the default values of all NTRs are "0", so no event will be generated.

Event Register

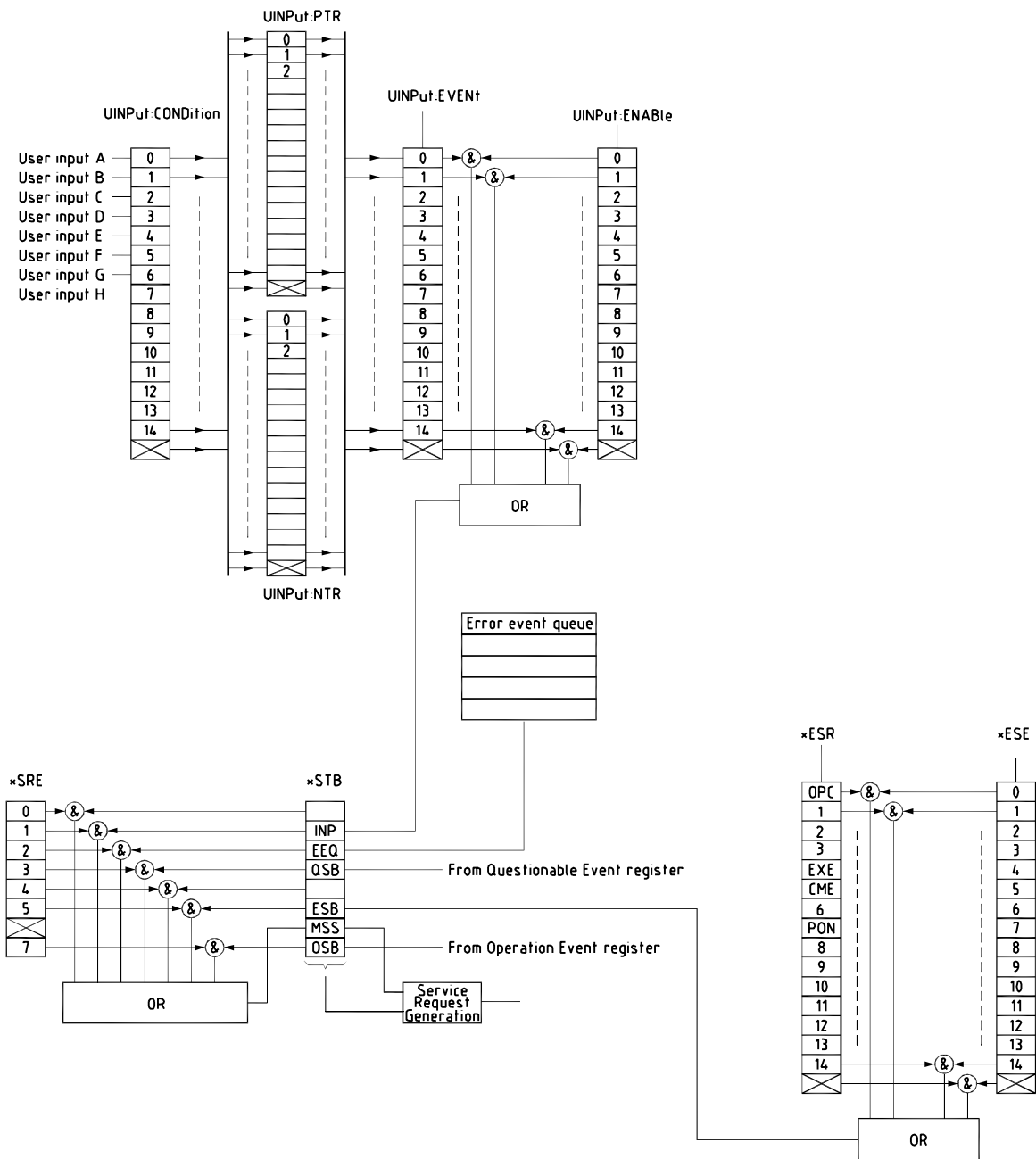
These registers capture changes in conditions (positive or negative transition, depending on the setting of PTR and NTR) at the corresponding condition register. This guarantees that the application can't miss an event that occurs.

Reading an event register will automatically clear the bits.

Enable Register

All bits in an event register can be summarized (OR-function) to a "summary bit", which is used in the next condition register higher in the structure. By setting the Enable register, the user can select which event bits in the corresponding event register will cause the summary bit.

After power-on the default values of all enable registers are "0".



***ESR?**

This command reads the Event Status Register and clears it afterwards.

This register is 16 bits long and contains the following status:

- bit 0 : OPC = stands for Operation Complete. This bit is set when a sequence is stopped.
- bit 4 : EXE = stands for Execution Error. This bit is set when the parameter of a command is outside the input range.
- bit 5 : CME = stands for CoMmand Error. This bit is set when an unknown command is received.
- bit 7 : PON = stands for Power On. This bit indicates that an off-to-on transistion has occurred in the supply.

The other bits are unused.

Syntax : ***ESR?<nl>**

***ESE**

This commands sets the 16 bit Event Status Enable register. This register determines which events from the ESR are summarized in bit 5 (ESB) of the Status Byte (STB) register.

A bit value 1 in the ESE register selects the corresponding event bit in the ESR to be reported in the STB register, bit 5.

The default value is "0".

Syntax : ***ESE<sp><NR1><nl>**

NR1 = 0 to 65535

***ESE?**

Reads-back the settings of the Event Status Enable register (ESE).

Syntax : ***ESE?<nl>**

***STB?**

This query reports the contents of the Status Byte register. The register contains the summary status of all overlaying registers and status queues. Reading the status byte with the *STB? query does not affect its contents. The bits will be cleared when the connected status registers are cleared, or with *CLS.

Bit	Name		Description Status Byte Register
0			Not used
1	INP	Input Summary Bit	Contains the summary bit of the User Input Register
2	EEQ	Error Event Queue	Set to 1 if there are error messages in the error queue
3	QSB	Questionable Summary Bit	Contains the summary of the Questionable Status Register.
4			Not used
5	ESB	Event Summary Bit	Contains the summary of the Event Status register.
6	MSS	Master Summary Status.	Set to 1 if one of the other bits of STB is set and enabled by the SRE register. This bit generates an SRQ.
7	OSB	Operation Summary Bit	Contains the summary of the Operation Event register

The Service Request Enable Register allows to select particular bits from the Status Byte to generate a Service Request (SRQ). This SRQ is send via a UDP message, port 8462 if a connection is made.

Syntax : ***STB?<nl>**

***SRE**

This Service Request Enable register determines which bits of the STB generate an SRQ.

The default value is "0".

Syntax : ***SRE<sp><NR1><nl>**

NR1 = 0 to 255

***SRE?**

This query reads the settings of the SRE register.

Syntax : ***SRE?<nl>**

***CLS**

This command clears all the "Event Registers".

The error queue is also cleared, so the red error LED will turn off after this command.

Syntax : ***CLS<nl>**

User Input Registers***Condition***

Syntax : **UINPut:CONDition?<nl>**

PTR

Syntax : **UINPut:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **UINPut:PTR?<nl>**

NTR

Syntax : **UINPut:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **UINPut:NTR?<nl>**

Event

Syntax : **UINPut:EVENT?<nl>**

Enable

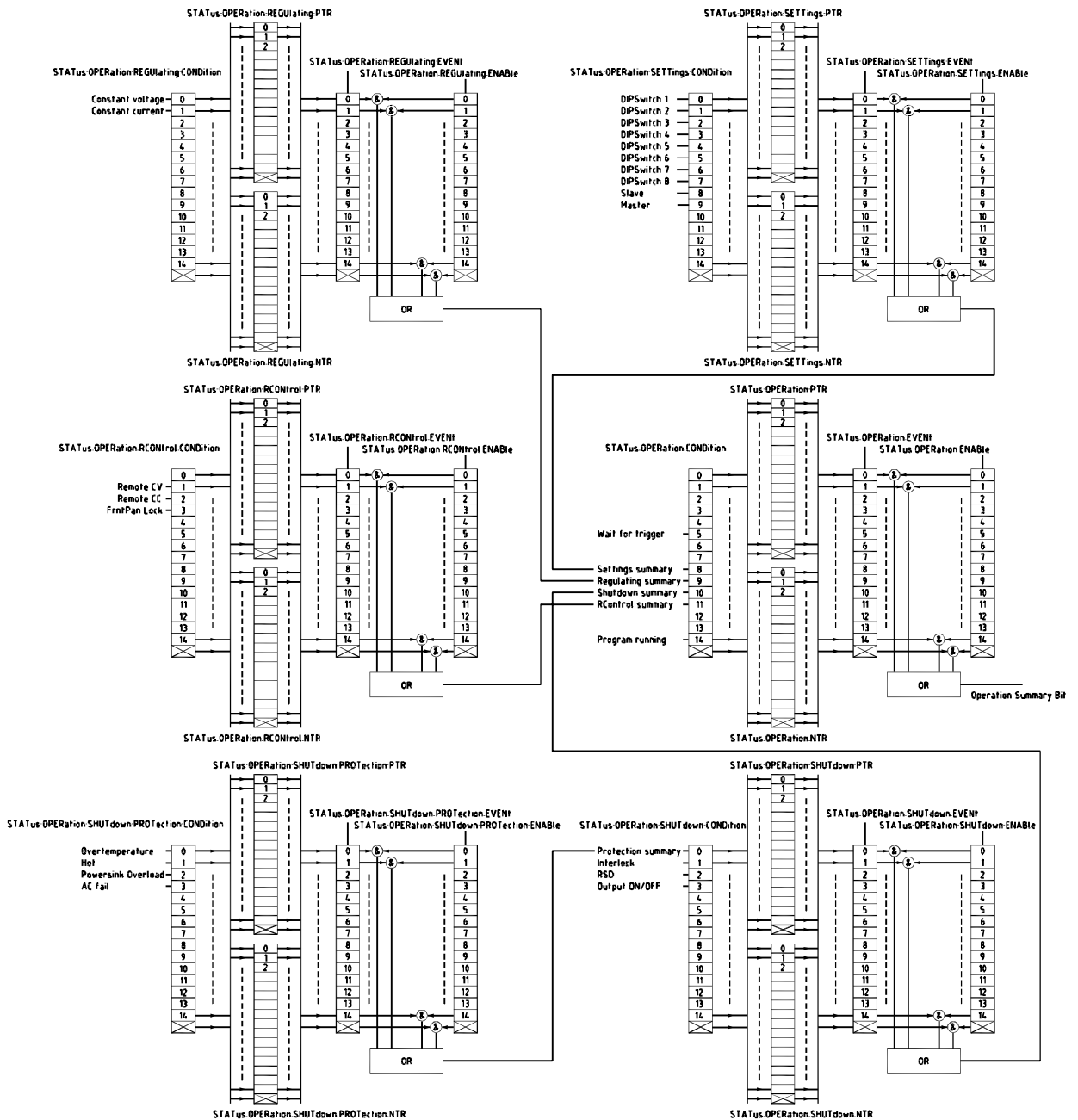
Syntax : **UINPut:ENABLE<sp><NR1><nl>**

To read the last settings:

Syntax : **UINPut:ENABLE?<nl>**

5.9 Status Operation Register Set

Status Operation Registers



Condition

Syntax : `STATUS:OPERation:CONDition?<nI>`

PTR

Syntax : `STATUS:OPERation:PTR<SP><NR1><nI>`

To read the last settings:

Syntax : `STATUS:OPERation:PTR?<nI>`

NTR

Syntax : **STATus:OPERation:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:NTR?<nl>**

Event

Syntax : **STATus:OPERation:EVENT?<nl>**

Enable

Syntax : **STATus:OPERation:ENABLE<sp><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:ENABLE?<nl>**

Status Operation Settings Registers**Confddition**

Syntax : **STATus:OPERation:SETTings:CONDition?<nl>**

PTR

Syntax : **STATus:OPERation:SETTings:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:SETTings:PTR?<nl>**

NTR

Syntax : **STATus:OPERation:SETTings:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:SETTings:NTR?<nl>**

Event

Syntax : **STATus:OPERation:SETTings:EVENT?<nl>**

Enable

Syntax : **STATus:OPERation:SETTings:ENABLE<sp><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:SETTings:ENABLE?<nl>**

Status Operation Regulating Registers**Condition**

Syntax : **STATus:OPERation:REGUlating:CONDition?<nl>**

PTR

Syntax : **STATus:OPERation:REGUlating:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:REGUlating:PTR?<nl>**

NTR

Syntax : **STATus:OPERation:REGUlating:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:REGUlating:NTR?<nl>**

Event

Syntax : **STATus:OPERation:REGUlating:EVENT?<nl>**

Enable

Syntax : **STATus:OPERation:REGUlating:ENABle**<sp><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:REGUlating:ENABle?**<nl>

Status Operation RControl Registers**Condition**

Syntax : **STATus:OPERation:RCONtrol:CONDition?**<nl>

PTR

Syntax : **STATus:OPERation:RCONtrol:PTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:RCONtrol:PTR?**<nl>

NTR

Syntax : **STATus:OPERation:RCONtrol:NTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:RCONtrol:NTR?**<nl>

Event

Syntax : **STATus:OPERation:RCONtrol:EVENT?**<nl>

Enable

Syntax : **STATus:OPERation:RCONtrol:ENABle**<sp><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:RCONtrol:ENABle?**<nl>

Status Operation Shutdown Registers**Condition**

Syntax : **STATus:OPERation:SHUTdown:CONDition?**<nl>

PTR

Syntax : **STATus:OPERation:SHUTdown:PTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:PTR?**<nl>

NTR

Syntax : **STATus:OPERation:SHUTdown:NTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:NTR?**<nl>

Event

Syntax : **STATus:OPERation:SHUTdown:EVENT?**<nl>

Enable

Syntax : **STATus:OPERation:SHUTdown:ENABle**<sp><NR1><nl>

To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:ENABle?**<nl>

Status Operation Shutdown Protection Registers**Condition**

Syntax : **STATus:OPERation:SHUTdown:PROTectio:n:CONDition?**<nl>

PTR

Syntax : **STATus:OPERation:SHUTdown:PROTection:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:PROTection:PTR?<nl>**

NTR

Syntax : **STATus:OPERation:SHUTdown:PROTection:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:PROTection:NTR?<nl>**

Event

Syntax : **STATus:OPERation:SHUTdown:PROTection:EVENT?<nl>**

Enable

Syntax : **STATus:OPERation:SHUTdown:PROTection:ENABLE<sp><NR1><nl>**

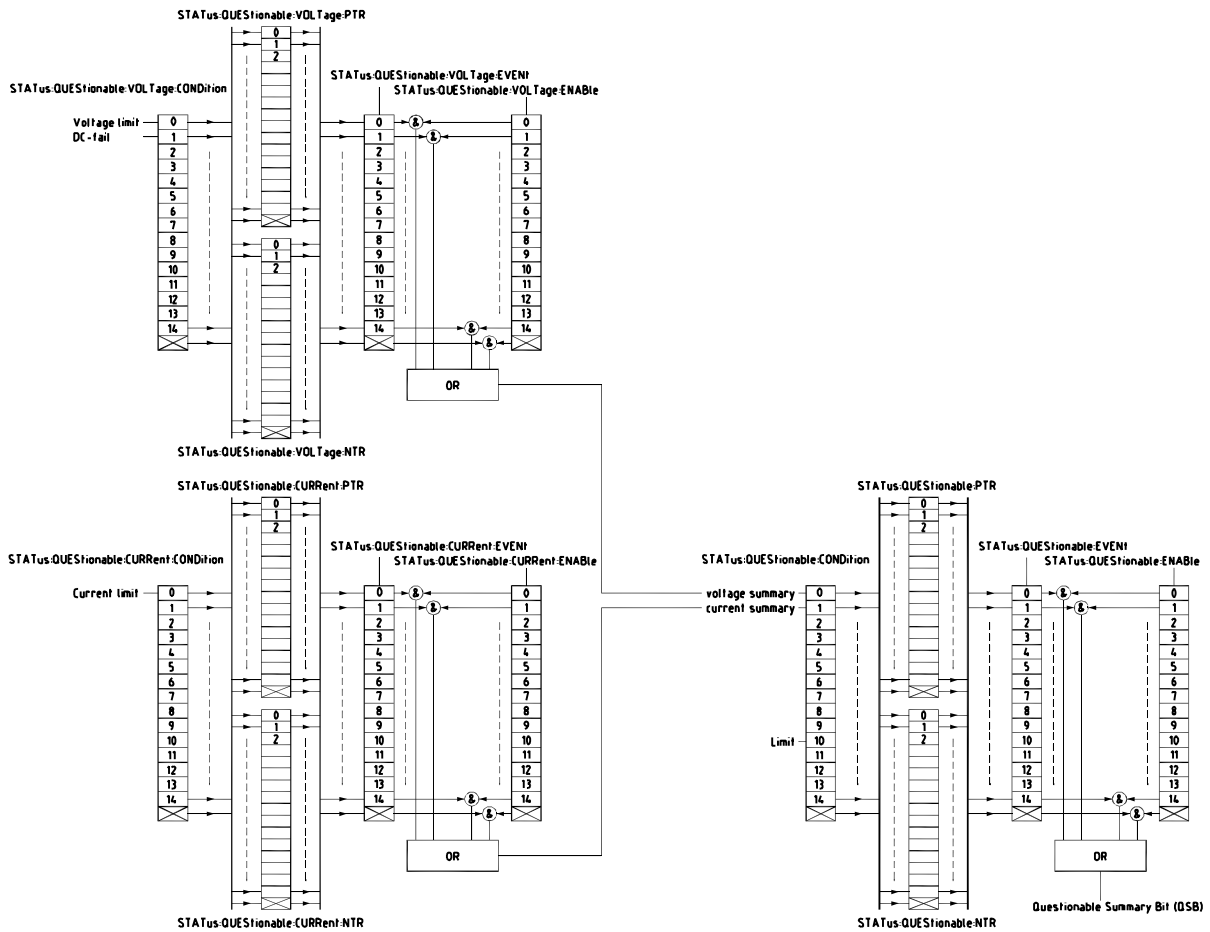
To read the last settings:

Syntax : **STATus:OPERation:SHUTdown:PROTection:ENABLE?<nl>**

5.10 Status Questionable Register Set

The questionable registers contain bits which give an indication of the quality of various status of the power supply.

Status Questionable Registers



Condition

Syntax : **STATus:QUESTIONable:CONDition?**<nl>

PTR

Syntax : **STATus:QUESTIONable:PTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:QUESTIONable:PTR?**<nl>

NTR

Syntax : **STATus:QUESTIONable:NTR**<SP><NR1><nl>

To read the last settings:

Syntax : **STATus:QUESTIONable:NTR?**<nl>

Event

Syntax : **STATus:QUESTIONable:EVENT?**<nl>

Enable

Syntax : **STATus:QUESTIONable:ENABLE**<sp><NR1><nl>

To read the last settings:

Syntax : **STATus:QUESTIONable:ENABLE?**<nl>

Status Questionable Voltage Registers

Condition

Syntax : **STATus:QUESTIONable:VOLTage:CONDition?<nl>**

PTR

Syntax : **STATus:QUESTIONable:VOLTage:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:VOLTage:PTR?<nl>**

NTR

Syntax : **STATus:QUESTIONable:VOLTage:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:VOLTage:NTR?<nl>**

Event

Syntax : **STATus:QUESTIONable:VOLTage:EVENT?<nl>**

Enable

Syntax : **STATus:QUESTIONable:VOLTage:ENABLE<sp><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:VOLTage:ENABLE?<nl>**

Status Questionable Current Registers

Condition

Syntax : **STATus:QUESTIONable:CURRent:CONDition?<nl>**

PTR

Syntax : **STATus:QUESTIONable:CURRent:PTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:CURRent:PTR?<nl>**

NTR

Syntax : **STATus:QUESTIONable:CURRent:NTR<SP><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:CURRent:NTR?<nl>**

Event

Syntax : **STATus:QUESTIONable:CURRent:EVENT?<nl>**

Enable

Syntax : **STATus:QUESTIONable:CURRent:ENABLE<sp><NR1><nl>**

To read the last settings:

Syntax : **STATus:QUESTIONable:CURRent:ENABLE?<nl>**

6 Sequencer

6.1 Introduction

The PSC-ETH 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 6.4). That makes it possible to run stand-alone, so no PC or network is required.

A sequence can set the output voltage / current, 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 6.5. 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.

Step numbers must start at 1, then 2, 3, 4, etc. It is not allowed to skip step numbers. If void steps are included for future implementations, they can be filled with the command NOP.

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
- **Ox**
O stands for user Output. X can be A to F (output A to F). This command can set or reset an digital output.
Syntax : **Ox=<boolean>** <boolean> = 0 or 1 x = A,B,C,D,E or F
- **#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,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

- **#K**
Variable K (#K) gives the opportunity to block(disable) or unblock (enable) the TCP/IP connection within the sequence.
#K=1 blocks the communication (which is in fact equal to removing the RJ45 cable physically). Also the ACT and LNK LEDs will be off. In this case the sequencer can't be interrupted by commands or broadcasts coming via the network.
#K=0 unblocks the connection and the LEDs will function again.
As soon as a sequence stops, the PSC-ETH automatically sets #K back to 0, so communication is possible again.
Syntax : **#K=<boolean>** <boolean> = 0 or 1.

Jumps

- **JP**
JP stands for Jump. It's an unconditional jump to a step anywhere in the sequence.
Syntax : **JP<sp><step>** <step> = any step number.
- **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 4 jumps.
Do not use RET without JS or visa versa. The sequencer will be in an undefined state.
Syntax : **JS<sp><step>** <step> = any step number.
RET
- **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>,<boolean>,<step>** x = input A,B,C,D,E,F,G or H
CJE<sp><Ox>,<boolean>,<step> x = output A,B,C,D,E or F
CJE<sp><#x>,<NR1>,<step> 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>,<boolean>,<step>** x = input A,B,C,D,E,F,G or H
CJNE<sp><Ox>,<boolean>,<step> x = output A,B,C,D,E or F
CJNE<sp><#x>,<NR2>,<step> 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 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 then the second) to the step declared in the third operand.
Syntax : **CJG<sp><SV>,<NR2>,<step>**
CJG<sp><MV>,<NR2>,<step>
CJG<sp><SC>,<NR2>,<step>
CJG<sp><MC>,<NR2>,<step>
CJG<sp><#x>,<NR1>,<step> 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, which is the current setting.
MV stands for Measure Volt, which is the actual output voltage.
MC stands for Measure Current, which is the actual output current.

For example CJK<sp>MV,10,35 ; When the actual output voltage is greater than 10V, the program jumps to step 35, 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 and the variables. CJK 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 then the second to the step declared in the third operand.

Syntax : **CJL<sp><SV>,<NR2>,<step>**
CJL<sp><MV>,<NR2>,<step>
CJL<sp><SV>,<NR2>,<step>
CJL<sp><SC>,<NR2>,<step>
CJL<sp><#x>,<NR1>,<step>

x = variable A,B,C,D,E,F,G,H,I or J

For example CJL<sp>SC,10,35 ; When the current setting is less than 10A, the program jumps to step 35, otherwise it continues with the next step.

Arithmetic

- **INC**

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

Syntax : **INC<sp><SV>,<NR2>**
INC<sp><SC>,<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 and Variables.

Syntax : **DEC<sp><SV>,<NR2>**
DEC<sp><SC>,<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 two 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

The deviation of this wait instruction is approximately 125µs, due to the communication process if there is still a connection. To decrease the deviation to 20µs, disconnect the PSC-ETH from the network. Within a sequence this can be done by a command. Refer to "Settings, #K" of this chapter.

- **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:operation:condition*-register (refer to chapter 5.9)

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**

6.3 Sequence control by commands

- **Read the Catalog**

The catalog contains all the programmed sequences. To read the catalog send the query:

PROG:CATalog?<nl>

The PSC-ETH 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 PSC-ETH only returns one <nl>.

Example : *PROG:CAT?<nl>* answer: *WAVE1<nl>PROCESS4<nl>RAMP-UP<nl><nl>*

- **How to select a Sequence**

To select an existing sequence or to create a new one, send the command:

PROG:SElected:NAME<sp><string><nl> string can be any character (max 16)

To read which sequence is selected, send the query:

PROG:SElected:NAME?<nl>

The PSC-ETH returns the name of the selected sequence, followed by a <nl>. Or, in case of no selection, only <nl>.

Sequence names are not case-sensitive.

- **Upload a Sequence to PSC-ETH (PC → PSC)**

First select/create a sequence, then upload the new steps by the command:

PROG:SElected:STEP<sp><NR1><sp><command+operand(s)><nl>

<NR1> = 1 to 2000. First program step 1, then step 2, etc. Existing steps will be overwritten.

For example : *PROG:SEL:STEP<sp>21<sp>CJNE<SP>#A,3,15<nl>*

- **Download a Sequence from PSC-ETH (PSC → PC)**

After a sequence is selected, the steps can be downloaded one by one, using the query:

PROG:SElected:STEP<sp><NR1>?<nl>

The PSC-ETH returns *<step number><sp><command+operand(s)><nl>*.

If the queried step doesn't exist, the PSC-ETH returns <nl>.

The PSC-ETH returns the complete sequence when it receives the query:

PROG:SElected:STEP<sp>?<nl>

The answer from the PSC-ETH will be:

<1><sp><command+operand(s)><nl><2><sp><command+operand(s)><nl> and so on.

After the last step the PSC-ETH sends another <nl> as a terminator.

- **Delete a Sequence**

After a sequence is selected, it can be removed from the catalog by:

PROG:SElected:DElete<nl>

If the sequence is in running mode, this command will stop the sequence immediately and erases it from memory.

To clear the whole catalog and delete all the sequences, including their assignments, send:

PROG:CATalog:DElete<nl> (available from V2.2)

- **Start a Sequence**

If a sequence is selected, it can be started by the command:

PROG:SElected:STATe<sp>RUN<nl>

The sequence will start at step 1.

- **Pause a Sequence**

If a sequence runs, it can be paused by the command:

PROG:SElected:STATe<sp>PAUSe<nl>

If the sequence is paused, it can be continued after sending the command:

PROG:SElected:STATe<sp>CONTInue<nl>

- **Step through a Sequence**

If a sequence is selected, it is possible to manually step through the program (during any mode) by:

PROG:SElected:STATe<sp>NEXT<nl>

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 example :

```
....
6 oa=1
7 w=100
8 ob=1
....
```

If the program executes step 7 to wait 100 seconds, the sequencer goes to step 8 after the PSC-ETH received the command PROG:SEL:STATE 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 oa=1
12 nop
13 cjne ia,1,12
14 oa=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.

For debugging the sequence this command can be used.

- **Stop a Sequence**

If the sequence mode is RUN or PAUSE , it can be stopped by the command:

PROG:SElected:STATe<sp><STOP><nl>

The sequence will stop immediately, but is still selected. The PSC-ETH restores the setting for voltage and current which were used before the sequence was started.

- **Read Sequence mode**

By sending the query:

PROG:SElected:STATe?<nl>

The PSC-ETH will return the current mode. There are three possibilities:

```
STOP<nl>
PAUSE,<step><nl>
RUN,<step><nl>
```

In RUN and PAUSE mode the returned answer also contains the step to be executed, separated by a comma.

- **Trigger a Step**

When a step contains TRG, the sequence waits for the command via TCP/IP:

TRIGger:IMMediate<nl>

After this command, the sequencer will proceed.

6.4 Sequence control by user inputs

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. So the power supply is able to work stand-alone in the field and control up to 8 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). 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 PSC-ETH restores the voltage and current settings that were valid before the sequence started.
- H** (Hold) ; When the sequence stops, the actual voltage and current settings remain the same.

Assignment examples:

- <seq name>**+ASR** ; The sequence starts when user input A becomes "1", it stops immediately when user input A becomes "0" and the voltage and current settings are restored.
- <seq name>**+DFH** : The sequence starts when user input D becomes "1". When D becomes "0", it finishes the steps until END is executed and holds the actual setting for voltage and current.

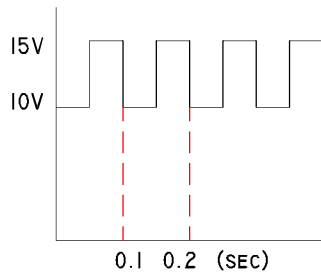
Any combination of the user inputs and the options can be made. It is required to set every character of the assignment. Refer to section 6.3 to find the command to program the sequence name. The length of the sequence name + the assignment has a maximum of 16 characters.

6.5 Sequence examples

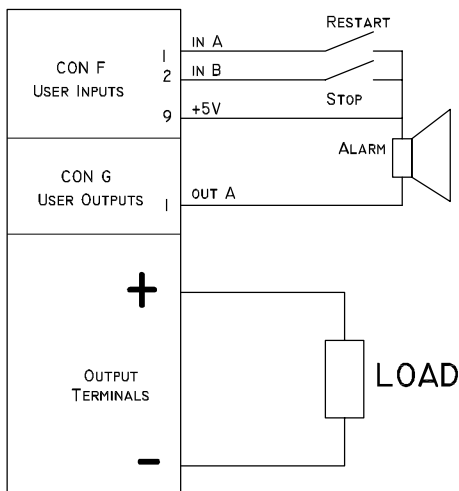
In this paragraph a few examples are explained based on typical applications. Every example contains a short description of the application, a flowchart, the list of the required sequence steps and the commands how to upload the sequence into the PSC-ETH.

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 oa=0
4 w=1
5 sv=10
6 w=0.05
7 sv=15
8 w=0.05
9 cje ib,1,16
10 cjc mc,26,5
11 sc=0
12 sv=0
13 oa=1
14 cjne ia,1,14
15 jp 3
16 sv=0
17 sc=0
18 end

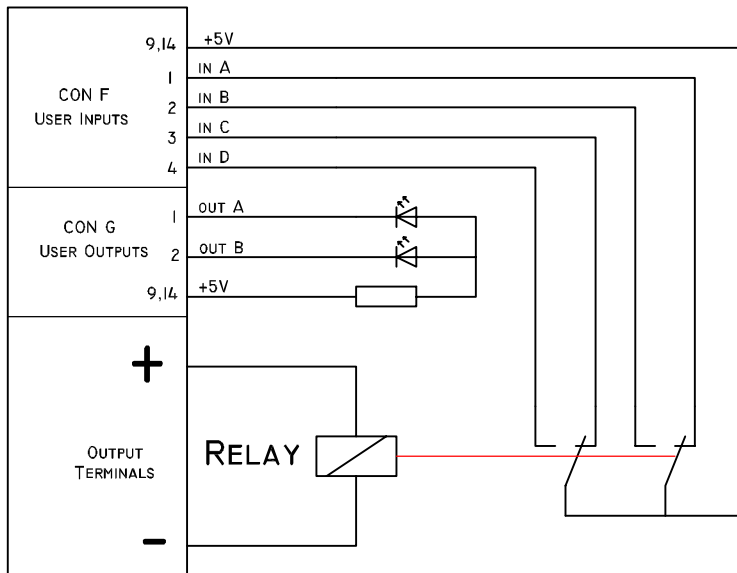
```

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 js 20
2 nop
3 cjl mc,0.01,4
4 w=1
5 sv=5.9
6 cjne ia,1,30
7 cjne ib,0,30
8 cjne ic,1,30
9 cjne id,0,30
10 cjl sv,11.8,30
11 inc sv,0.05
12 w=0.02
13 cjne ib,1,10
14 cjne ia,0,30
15 cjne ic,0,30
16 cjne id,1,30
17 ob=1
18 jp 1
19 end
20 sv=5
21 sc=0.03
22 cjl mc,0.01,22
23 oa=0
24 ob=0
25 nop
26 nop
27 nop
28 nop
29 ret
30 oa=1
31 jp 1
32 nop
33 nop

```

7 Command list TCP/IP

Index TCP / IP

*CLS<nl>	page 5-14
*ESE?<nl>	page 5-13
*ESE<sp><NR1><nl>	page 5-13
*ESR?<nl>	page 5-13
*IDN?<nl>.	page 5-1
*PUD?<nl>	page 5-1
*PUD<sp><data><nl>	page 5-1
*RCL<nl>	page 5-2
*SAV<nl>	page 5-1
*SAV<sp><password><nl>	page 5-1
*SRE?<nl>	page 5-14
*SRE<sp><NR1><nl>	page 5-13
*STB?<nl>	page 5-13
CAL<sp>0,<NR2><nl>.	page 5-6
CAL<sp>0?<nl>	page 5-6
CAL<sp>1,<NR2><nl>.	page 5-6
CAL<sp>1?<nl>	page 5-6
CAL<sp>2,<NR2><nl>.	page 5-6
CAL<sp>2?<nl>	page 5-6
CAL<sp>3,<NR2><nl>.	page 5-6
CAL<sp>3?<nl>	page 5-6
CAL<sp>4,<NR2><nl>.	page 5-6
CAL<sp>4?<nl>	page 5-6
CAL<sp>5,<NR2><nl>.	page 5-6
CAL<sp>5?<nl>	page 5-6
CAL<sp>6,<NR2><nl>.	page 5-6
CAL<sp>6?<nl>	page 5-6
CAL<sp>7,<NR2><nl>.	page 5-6
CAL<sp>7?<nl>	page 5-6
MEASure:CURRent?<nl>	page 5-4
MEASure:POWer?<nl>	page 5-4
MEASure:VOLTagE?<nl>	page 5-4
OUTPut?<nl>.	page 5-10
OUTPut<sp><boolean><nl>	page 5-10
PROGram:CATalog:DELeTe<nl>	page 6-4
PROGram:CATalog?<nl>	page 6-4
PROGram:SELeCted:DELeTe<nl>.	page 6-4
PROGram:SELeCted:NAME?<nl>	page 6-4
PROGram:SELeCted:NAME<sp><string><nl>.	page 6-4
PROGram:SELeCted:STATe?<nl>	page 6-5
PROGram:SELeCted:STATe<sp>CONTInue<nl>	page 6-4
PROGram:SELeCted:STATe<sp>NEXT<nl>	page 6-4
PROGram:SELeCted:STATe<sp>PAUSe<nl>	page 6-4
PROGram:SELeCted:STATe<sp>RUN<nl>	page 6-4
PROGram:SELeCted:STATe<sp>STOP<nl>	page 6-5
PROGram:SELeCted:STEP<sp><NR1>?<nl>	page 6-4
PROGram:SELeCted:STEP<sp><NR1><sp><command+operand(s)><nl>	page 6-4
PROGram:SELeCted:STEP<sp>?<nl>	page 6-4
SOURce:CURRent:MAXimum?<nl>	page 5-3

SOURce:CURRent:MAXimum<sp><NR2><nl>	page 5-3
SOURce:CURRent?<nl>	page 5-3
SOURce:CURRent<sp><NR2><nl>	page 5-3
SOURce:VOLTagE:MAXimum?<nl>	page 5-3
SOURce:VOLTagE:MAXimum<sp><NR2><nl>	page 5-3
SOURce:VOLTagE?<nl>	page 5-3
SOURce:VOLTagE<sp><NR2><nl>	page 5-3
STATus:OPERation:CONDition?<nl>	page 5-15
STATus:OPERation:ENABLE?<nl>	page 5-16
STATus:OPERation:ENABLE<sp><NR1><nl>	page 5-16
STATus:OPERation:EVENT?<nl>	page 5-16
STATus:OPERation:NTR?<nl>	page 5-16
STATus:OPERation:NTR<sp><NR1><nl>	page 5-16
STATus:OPERation:PTR?<nl>	page 5-15
STATus:OPERation:PTR<sp><NR1><nl>	page 5-15
STATus:OPERation:RCONtrol:CONDition?<nl>	page 5-17
STATus:OPERation:RCONtrol:ENABLE?<nl>	page 5-17
STATus:OPERation:RCONtrol:ENABLE<sp><NR1><nl>	page 5-17
STATus:OPERation:RCONtrol:EVENT?<nl>	page 5-17
STATus:OPERation:RCONtrol:NTR?<nl>	page 5-17
STATus:OPERation:RCONtrol:NTR<sp><NR1><nl>	page 5-17
STATus:OPERation:RCONtrol:PTR?<nl>	page 5-17
STATus:OPERation:RCONtrol:PTR<sp><NR1><nl>	page 5-17
STATus:OPERation:REGULating:CONDition?<nl>	page 5-16
STATus:OPERation:REGULating:ENABLE?<nl>	page 5-17
STATus:OPERation:REGULating:ENABLE<sp><NR1><nl>	page 5-17
STATus:OPERation:REGULating:EVENT?<nl>	page 5-16
STATus:OPERation:REGULating:NTR?<nl>	page 5-16
STATus:OPERation:REGULating:NTR<sp><NR1><nl>	page 5-16
STATus:OPERation:REGULating:PTR?<nl>	page 5-16
STATus:OPERation:REGULating:PTR<sp><NR1><nl>	page 5-16
STATus:OPERation:SETTings:CONDition?<nl>	page 5-16
STATus:OPERation:SETTings:ENABLE?<nl>	page 5-16
STATus:OPERation:SETTings:ENABLE<sp><NR1><nl>	page 5-16
STATus:OPERation:SETTings:EVENT?<nl>	page 5-16
STATus:OPERation:SETTings:NTR?<nl>	page 5-16
STATus:OPERation:SETTings:NTR<sp><NR1><nl>	page 5-16
STATus:OPERation:SETTings:PTR?<nl>	page 5-16
STATus:OPERation:SETTings:PTR<sp><NR1><nl>	page 5-16
STATus:OPERation:SHUTdown:CONDition?<nl>	page 5-17
STATus:OPERation:SHUTdown:ENABLE?<nl>	page 5-17
STATus:OPERation:SHUTdown:ENABLE<sp><NR1><nl>	page 5-17
STATus:OPERation:SHUTdown:EVENT?<nl>	page 5-17
STATus:OPERation:SHUTdown:NTR?<nl>	page 5-17
STATus:OPERation:SHUTdown:NTR<sp><NR1><nl>	page 5-17
STATus:OPERation:SHUTdown:PROTectioN:CONDition?<nl>	page 5-17
STATus:OPERation:SHUTdown:PROTectioN:ENABLE?<nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:ENABLE<sp><NR1><nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:EVENT?<nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:NTR?<nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:NTR<sp><NR1><nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:PTR?<nl>	page 5-18
STATus:OPERation:SHUTdown:PROTectioN:PTR<sp><NR1><nl>	page 5-18
STATus:OPERation:SHUTdown:PTR?<nl>	page 5-17

STATus:OPERation:SHUTdown:PTR<sp><NR1><nl>	page 5-17
STATus:QUEStionable:CONDition?<nl>	page 5-19
STATus:QUEStionable:CURRent:CONDition?<nl>	page 5-20
STATus:QUEStionable:CURRent:ENABLE?<nl>	page 5-20
STATus:QUEStionable:CURRent:ENABLE<sp><NR1><nl>	page 5-20
STATus:QUEStionable:CURRent:EVENT?<nl>	page 5-20
STATus:QUEStionable:CURRent:NTR?<nl>	page 5-20
STATus:QUEStionable:CURRent:NTR<sp><NR1><nl>	page 5-20
STATus:QUEStionable:CURRent:PTR?<nl>	page 5-20
STATus:QUEStionable:CURRent:PTR<sp><NR1><nl>	page 5-20
STATus:QUEStionable:ENABLE?<nl>	page 5-19
STATus:QUEStionable:ENABLE<sp><NR1><nl>	page 5-19
STATus:QUEStionable:EVENT?<nl>	page 5-19
STATus:QUEStionable:NTR?<nl>	page 5-19
STATus:QUEStionable:NTR<sp><NR1><nl>	page 5-19
STATus:QUEStionable:PTR?<nl>	page 5-19
STATus:QUEStionable:PTR<sp><NR1><nl>	page 5-19
STATus:QUEStionable:VOLTag:e:CONDition?<nl>	page 5-20
STATus:QUEStionable:VOLTag:e:ENABLE?<nl>	page 5-20
STATus:QUEStionable:VOLTag:e:ENABLE<sp><NR1><nl>	page 5-20
STATus:QUEStionable:VOLTag:e:EVENT?<nl>	page 5-20
STATus:QUEStionable:VOLTag:e:NTR?<nl>	page 5-20
STATus:QUEStionable:VOLTag:e:NTR<sp><NR1><nl>	page 5-20
STATus:QUEStionable:VOLTag:e:PTR?<nl>	page 5-20
STATus:QUEStionable:VOLTag:e:PTR<sp><NR1><nl>	page 5-20
SYSTem:ERRor?<nl>	page 5-9
SYSTem:FRONTpanel :STATus?<nl>	page 5-8
SYSTem:FRONTpanel :STATus<sp><boolean><nl>	page 5-8
SYSTem:PASSword:STATus?<nl>	page 5-9
SYSTem:PASSword<sp><old_password>,<new_password><nl>	page 5-9
SYSTem:REMote :STATus?<nl>	page 5-8
SYSTem:REMote :STATus<sp><setting><nl>	page 5-8
SYSTem:REMote:CC :STATus?<nl>	page 5-8
SYSTem:REMote:CC :STATus<sp><setting><nl>	page 5-8
SYSTem:REMote:CV :STATus?<nl>	page 5-8
SYSTem:REMote:CV :STATus<sp><setting><nl>	page 5-8
SYSTem:RSD :STATus?<nl>	page 5-8
SYSTem:RSD :STATus<sp><boolean><nl>	page 5-8
TRIGger:IMMEDIATE<nl>	page 6-5
UINPut:CONDition?<nl>	page 5-7, page 5-14
UINPut:ENABLE?<nl>	page 5-14
UINPut:ENABLE<sp><NR1><nl>	page 5-14
UINPut:EVENT?<nl>	page 5-14
UINPut:NTR?<nl>	page 5-14
UINPut:NTR<sp><NR1><nl>	page 5-14
UINPut:PTR?<nl>	page 5-14
UINPut:PTR<sp><NR1><nl>	page 5-14
UOUTput?<nl>	page 5-7
UOUTput<sp><0+NR><nl>	page 5-7

8 Command list Sequencer

Index Sequencer

#I= <NR1>	page 6-1
#J= <NR1>	page 6-1
#K=<boolean>	page 6-2
#x= <NR1>	page 6-1
CJE<sp><#x>,<NR1>,<step>	page 6-2
CJE<sp><Ix>,<boolean>,<step>	page 6-2
CJE<sp><Ox>,<boolean>,<step>	page 6-2
CJG<sp><#x>,<NR1>,<step>	page 6-2
CJG<sp><MC>,<NR2>,<step>	page 6-2
CJG<sp><MV>,<NR2>,<step>	page 6-2
CJG<sp><SC>,<NR2>,<step>	page 6-2
CJG<sp><SV>,<NR2>,<step>	page 6-2
CJL<sp><#x>,<NR1>,<step>	page 6-3
CJL<sp><MV>,<NR2>,<step>	page 6-3
CJL<sp><SC>,<NR2>,<step>	page 6-3
CJL<sp><SV>,<NR2>,<step>	page 6-3
CJNE<sp><#x>,<NR2>,<step>	page 6-2
CJNE<sp><Ix>,<boolean>,<step>	page 6-2
CJNE<sp><Ox>,<boolean>,<step>	page 6-2
DEC<sp><#x>,<NR1>	page 6-3
DEC<sp><SC>,<NR2>	page 6-3
DEC<sp><SV>,<NR2>	page 6-3
END	page 6-3
INC<sp><#x>,<NR1>	page 6-3
INC<sp><SC>,<NR2>	page 6-3
INC<sp><SV>,<NR2>	page 6-3
JP<sp><step>	page 6-2
JS<sp><step>	page 6-2
NOP	page 6-3
Ox=<boolean>	page 6-1
RET	page 6-2
SC= <NR2>	page 6-1
SV= <NR2>	page 6-1
TRG	page 6-3
W= <NR2>	page 6-3