



PSC-ETH Gen2

- **PSC-ETH-2 INT (Internal interface Card)**
- **PSC-ETH-2 EXT (External Interface Module)**

Firmware Update

It is strongly recommended, first to perform a firmware update before further operation. See this manual for instructions.

Driver & Example Software

For several applications and Interfaces there is Driver & Example Software available on our website. See [PRODUCTS\PSC-ETH\DOWNLOADS](#).

PRODUCT MANUAL

Firmware version P0102

Contents:

- 1 – In General
- 2 – Installation
- 3 – Communication
- 4 – Downloads & Web Interface
- 5 – Conventions
- 6 – Command description
- 7 – Sequencer
- 8 – Command list TCP/IP
- 9 – Command list Sequencer
- 10 – EU Declaration

Table of Contents

1	In General	4
1.1	Features	4
1.2	Analog	4
1.3	Status	4
1.4	LEDs	4
1.5	Digital I/O	5
1.5.1	User Inputs (connector CON F):	5
1.5.2	User Outputs (connector CON G):	5
2	Installation	6
2.1	Infrastructure	6
2.2	Software	6
2.3	Settings	6
2.4	Terminal	7
2.5	DE Easy Control	7
3	Communication	8
3.1	Settings	8
3.2	TCP/IP	8
4	Downloads & Web Interface	9
4.1	Firmware updating	9
4.2	Firmware	9
4.3	Web Interface	9
4.3.1	Console	9
4.3.2	Configuration	10
4.3.3	Administration.....	11
4.3.4	Documentation	11
4.3.5	Ethernet.....	11
4.3.6	Sequencer.....	11
4.4	Forgotten password, access key or network settings	11
5	Conventions	12
5.1	Syntax	12
5.2	Query	12
5.3	Space <sp>	12
5.4	Terminator <term>	12
5.5	Parameters	12
6	Command description	13
6.1	General Instructions	13
6.1.1	*IDN?.....	13
6.1.2	*PUD	13
6.1.3	*SAV.....	13
6.1.4	*RST.....	13
6.1.5	*RCL.....	14
6.2	Source Subsystem	14
6.2.1	Introduction.....	14
6.2.2	Maximum Output Voltage	14
6.2.3	Maximum Output Current.....	14
6.2.4	Set Output Voltage	14
6.2.5	Set Output Current	14
6.2.6	Output Voltage Stepsize	14
6.2.7	Output Current Stepsize	14
6.3	Measure Subsystem	15
6.3.1	Introduction.....	15
6.3.2	Measure Output Voltage.....	15
6.3.3	Measure Output Current	15
6.3.4	Measure Output Power.....	15
6.4	Calibrate Subsystem	15
6.4.1	Introduction.....	15
6.4.2	Definitions.....	16
6.4.3	Set A: compatible with PSC ETH.....	16
6.4.4	Set B: compatible with PSC-ETH-2 / SM-Series	17
6.5	Digital User In-/Outputs	18
6.5.1	User Outputs	18
6.5.2	User Inputs	18
6.6	System Subsystem	18
6.6.1	Remote Shut Down (RSD).....	18
6.6.2	Limits.....	19
6.6.3	Front Panel Lock	19
6.6.4	Remote programming	19
6.6.5	Error Message.....	20
6.6.6	Password	20
6.6.7	Watchdog	20
6.6.8	Terminator.....	21
6.7	Output	22
6.8	Register Structure	22
6.8.1	*CLS.....	22
7	Sequencer	23
7.1	Introduction	23
7.2	Commands	23
7.2.1	Settings	23

7.2.2	Jumps.....	23
7.2.3	Arithmetic	24
7.2.4	Miscellaneous.....	25
7.3	Sequence control by commands	26
7.3.1	Read the Catalog.....	26
7.3.2	How to create or select a Sequence	26
7.3.3	Upload a Sequence to PSC-ETH-2 (PC → PS)	26
7.3.4	Download a Sequence from PSC-ETH-2 (PS → PC).....	26
7.3.5	Delete a Sequence	26
7.3.6	Start a Sequence.....	26
7.3.7	Pause a Sequence	26
7.3.8	Step through a Sequence	26
7.3.9	Stop a Sequence	27
7.3.10	Read Sequence mode	27
7.3.11	Trigger a Step.....	27
7.3.12	Add labels.....	27
7.3.13	Delete labels.....	27
7.3.14	Building a Sequence.....	27
7.3.15	Select saving to non-volatile memory.....	28
7.3.16	Saving to non-volatile memory.....	28
7.4	Sequence control by Web	28
7.4.1	Read the Catalog.....	28
7.4.2	How to create a sequence	29
7.4.3	How to select a sequence.....	29
7.4.4	Upload a sequence to PSC-ETH-2 (PC → PS).....	29
7.4.5	Download a Sequence from PSC-ETH-2 (PS → PC).....	30
7.4.6	Delete a Sequence	30
7.4.7	Start a Sequence.....	30
7.4.8	Pause a Sequence	30
7.4.9	Step through a Sequence	30
7.4.10	Stop a Sequence	30
7.4.11	Read Sequence mode	30
7.4.12	Trigger a Step.....	30
7.4.13	Add labels.....	30
7.4.14	Delete labels.....	30
7.4.15	Building a Sequence.....	30
7.4.16	Saving a sequence to non-volatile memory.....	31
7.5	Sequence control by user inputs	31
7.6	Sequence examples	32
7.6.1	Example 1: Generate waveform.....	32
7.6.2	Example 2: Test relays	33
8	Command list TCP/IP	34
8.1	Index TCP / IP	34
9	Command list Sequencer.....	36
9.1	Index Sequencer	36
10	EU-Declaration of Conformity - PSC-ETH-2.....	38

1 In General

1.1 Features

The PSC-ETH-2 is an interface between a TCP/IP network and a Power Supply. It allows the operator to create fully automated systems. The PSC-ETH-2 can set and measure the parameters of the power supply, read the status, set controls, interact 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 PSC-ETH-2 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

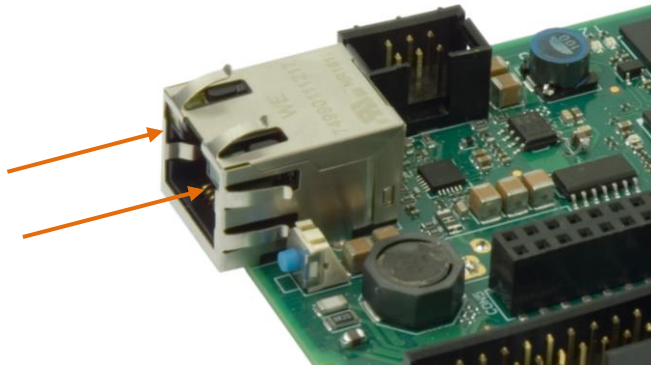
The power supplies are equipped with a lot of signals which inform the user about the status of the supply. Status like Limit, OverTemperature, DCF, etc. can be read to check the condition of the system. For example, ACF-signal (which indicates an incorrect input voltage) can be monitored, so action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

1.4 LEDs

Inside 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.



1.5 Digital I/O

The SM800, SM1500, SM6000 and the external PSC-ETH-2 provide 8 user inputs and 8 outputs. These can be controlled / monitored by commands (refer to chapter 5) or can be used to interact with the Sequencer (refer to chapter 6) to make the power supply act like a power PLC.

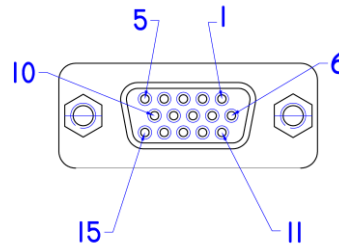
1.5.1 User Inputs (connector CON F):

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 2.5 V - 30 V and have an 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

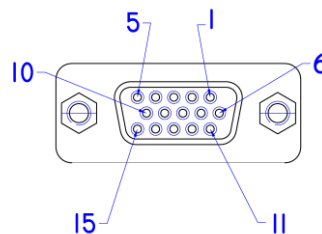


1.5.2 User Outputs (connector CON G):

The 8 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 = user output G
- 8 = user output H
- 9 = +5 V
- 10 = 0 V
- 11 = n.c.
- 12 = n.c.
- 13 = n.c.
- 14 = +5 V
- 15 = 0 V



2 Installation

Warning!

Carefully read the chapter "Safety Instructions" in the Delta Elektronika power supply manual before connecting the internal ETH Interface or the external ETH module!

With regards safety, the Ethernet connector, the User I/O connectors and the Analog Programming connector (only on the external module) are all at the level of the 'minus' DC power terminal of the unit. The power output is floating in relation to the Protective Earth and can be at a hazardous high voltage!

When the 'minus' DC power terminal of the unit can exceed 60Vdc / 42.4Vpk in respect to Protective Earth, additional external measures must be taken to ensure safety isolation PSC-Ethernet connections.

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 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"- 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>" can be used (e.g. via Windows© menu Start - RUN - CMD) to check the connection. For example: PING 10.1.0.101.

2.2 Software

On our [website](#), go to PRODUCTS > INTERFACES > PSC ETH INTERFACE > DOWNLOADS and download the "Software Ethernet Interface" for installation and applications.

2.3 Settings

Each network has its own range of IP addresses. To be able to control the PSC-ETH-2 via a particular network, the IP address of the PSC-ETH-2 must be within the address range of that network.

The setup installs a program named "PSC-ETH Configurator". This program shows the IP addresses of the PSC-ETH-2s 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-2.

By clicking on (one of) the PSC-ETH-2(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: 255.240.0.0)
Class C:	192.168.0.0	to	192.168.255.255	(mask: 255.255.0.0)

with the 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-2 will be out of range. That is why this program blocks these address settings.

When IP-settings are unknown, press the reset button next to the ETH connector. See paragraph 4.4 for details.

When more PSC-ETH-2 devices are used, it is recommended to give them specific names, using the command *PUD (refer to chapter 6.1).

2.4 Terminal

The setup also installs a program named "PSC-ETH Terminal". This program can be used to send commands to and read queries from the PSC-ETH-2 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-2, identified with an "auto query" LED on the screen after a command is send. The answer appears in the "Characters Read"-box.

2.5 DE Easy Control

During setup you can choose to install a program called "DE Easy Control". This is a graphical user interface that resembles the front panel of the power supply. The program can be used to set and monitor the output of the power supply.

By clicking the arrow button of the "unit" drop down menu in the upper right corner of the program a scan is started for any power supply with a PSC-ETH-2 connected to the PC. (The connection can be a direct link or via a network.) The result of the scan is shown in a list which contains the IP addresses, PUD strings, Serial numbers or MAC addresses of the interfaces. From this list a power supply can be selected. The identifier shown in the list (IP, PUD, Serial or MAC) depends on the selected option in the programs upper right corner.

The appearance of the user interface depends on the specifications and features of the selected power supply. This means that the scale of the voltage and current knobs will automatically adjust to the maximum allowed values stored in the PSC-ETH-2. The status indicators will appear or disappear automatically as well, depending on the features that the power supply offers.

The settings for both voltage and current can be changed by turning the knobs or by entering the required value in the boxes below the knobs. The measured value is shown in the displays above the knobs.

To run the program, the "LabVIEW Run-Time Engine 2015" is required. (Copyright © 2015 National Instruments Corporation. All rights reserved.) This Run-Time Engine is included in the downloadable software package on the website. It can also be downloaded from www.ni.com.

Disclaimer

This software is provided by Delta Elektronika B.V. "as is" without guarantee. The usage of this software is at own risk. In no event shall Delta Elektronika BV be liable for any damage as a result from the use, misuse, inability to use, faulty operation, installation or adjustments of the software. Delta Elektronika does not accept any responsibility with regards to losses of the owner or third party users as a result of the usage of this software.

3 Communication

3.1 Settings

There are two important settings to communicate properly with the PSC-ETH-2, 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-2. To show the user / programmer how to communicate with the PSC-ETH-2, examples of LabVIEW are provided on the [website](#).

Disclaimer

This software is provided by Delta Elektronika B.V. "as is" without guarantee. The usage of this software is at own risk. In no event shall Delta Elektronika BV be liable for any damage as a result from the use, misuse, inability to use, faulty operation, installation or adjustments of the software. Delta Elektronika does not accept any responsibility with regards to losses of the owner or third party users as a result of the usage of this software.

4 Downloads & Web Interface

4.1 Firmware updating

Check the version of the firmware in the unit via Menu > System Info > Unit > Version.
 Go to www.delta-elektronika.nl and check if there is new firmware available via Products -> PSC-ETH -> Downloads. Download the new firmware package to the computer.
 Connect the unit to the above computer via LAN and open the PSC-ETH-2 web interface using an internet browser.

The web interface is found by entering the IP-address of the unit in the address bar of the browser.
 Note: when DHCP is enabled the IP-address can change, for example after a power cycle.

In the web interface, go to Administration -> Firmware.
 Select "Choose File" and browse to the downloaded package, enter password if enabled and "Start Update". See below figure 4 - 1 for a screen shot of the web interface.
 Recommended firmware package is P0100.

***Note:** when DHCP is enabled, the IP-address can change, for example after a power cycle.

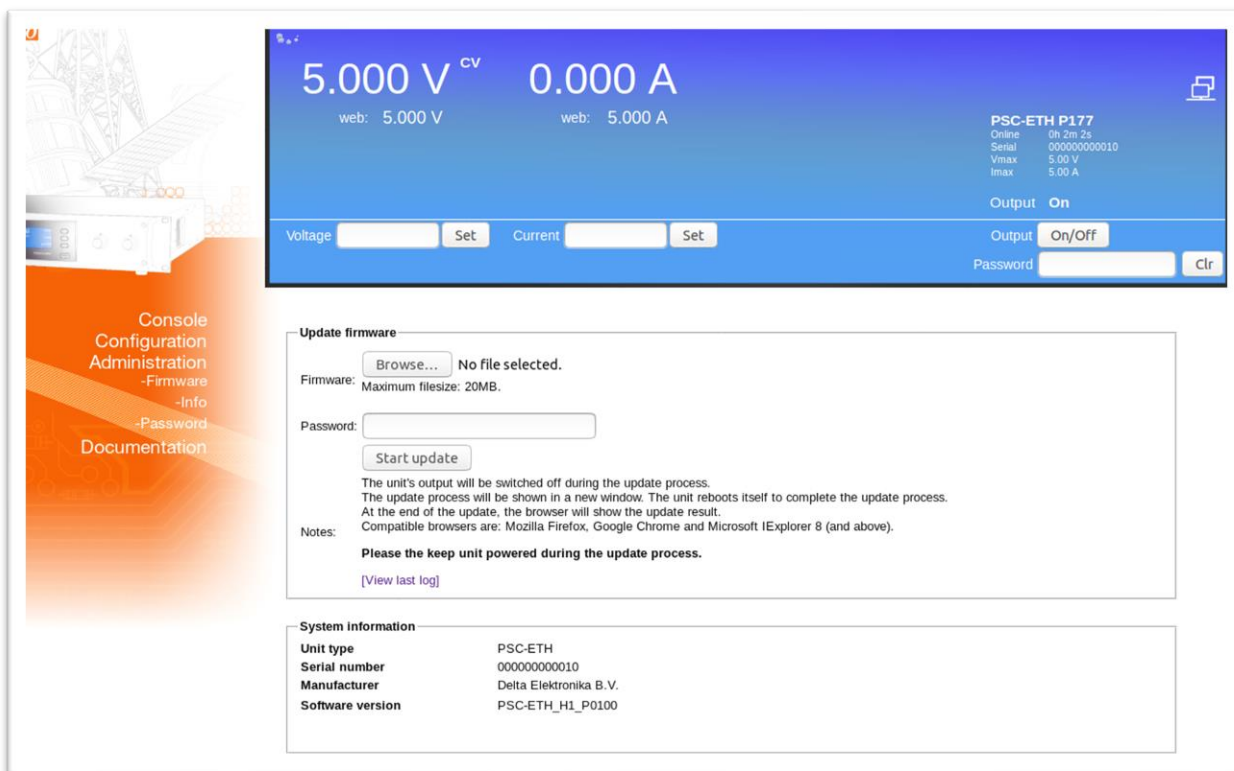


Fig. 4 - 1 - Regularly check for new versions of user manual and firmware.

4.2 Firmware

Check at www.delta-elektronika.nl for latest firmware via: Products > PSC-ETH > Downloads.

Note: This document is based on firmware version 0102. It is strongly recommended to regularly check for updates for additional functionality and improvements.

4.3 Web Interface

It is advised to use the web browsers Mozilla Firefox, Google Chrome or MS Edge or later.
 The web interface is available 15 seconds after start up of the unit.
 The below menu items are available in the web interface:

4.3.1 Console

4.3.1.1 Frontpanel

Possible settings via the console:
 - voltage and current
 - output On/Off

Possible monitoring via the console:

- actual and set values of voltage and current
- output setting (on/off)
- status icons, for example DC-fail
- type of unit and serial number

See fig. 4 - 2 for the console lay-out.

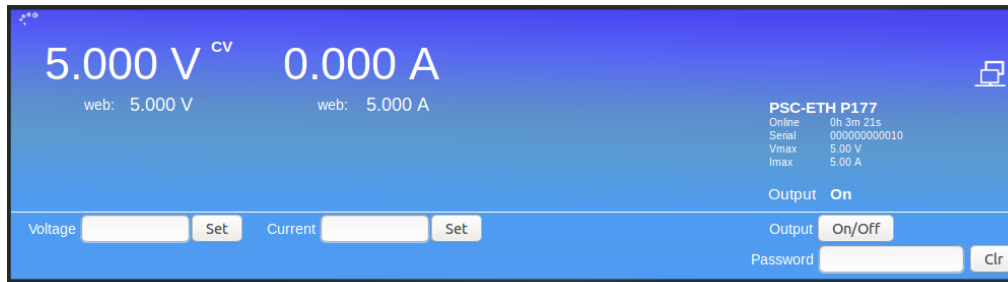


Fig. 4 - 2 - Front console for setting of the output and monitoring various parameters.

4.3.1.2 Sequencer

Possible to select sequences from the unit memory.

Running, Pausing and Stopping of sequences.

Trigger sequence

Running in Single Step mode.

Monitor sequencer variables and timers via the "advanced" button.

See fig 4 - 3 for the console lay-out.

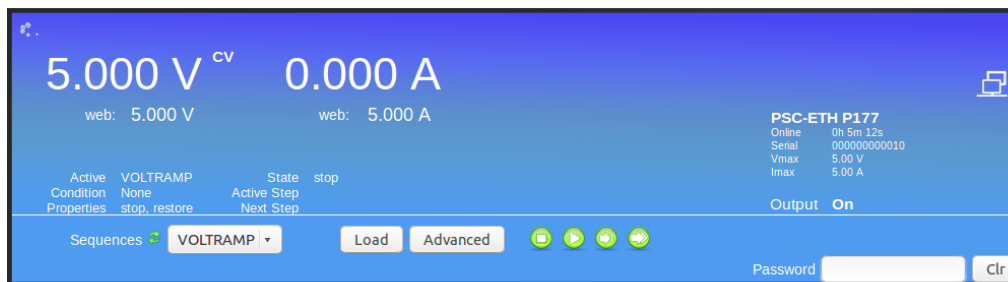


Fig. 4 - 3 - Sequencer console for selecting and controlling sequences.

4.3.2 Configuration

4.3.2.1 General

LIMITS

Switch Voltage Limit ON or OFF and set value.

Switch Current Limits ON or OFF and set values.

SOURCES

Set the program source for voltage control.

Set the program source for current control.

NETWORK

DHCP enabled / disabled.

IP Version.

Network IP address.

Network Subnet mask.

Network Gateway address.

Network interface MAC address.

SEQUENCES

Upload sequences into the unit's volatile memory.

Synchronize memory to copy sequences from the volatile to the non-volatile memory.

After switching off the unit, the sequences remain on the power supply.

Monitor and make settings:

- View sequencer name
- View if it is loaded as active sequencer
- View if it has been build

- Mark for Non-Volatile
- Set start/stop conditions
- Set if to restore or retain output state and values after it is terminated
- Mark for deletion

See paragraph "SEQUENCER" in this chapter for more information about sequencer programming.

4.3.3 Administration

4.3.3.1 Firmware

Here a new firmware package can be uploaded.

4.3.3.2 Info

System information

- Unit.
- Serial number.
- Manufacturer.
- Software version.

4.3.3.3 Password

Change the password to block the unit.

The default password is "depower".

Passwords are not case sensitive.

In case of a forgotten password see next paragraph in this chapter.

4.3.4 Documentation

Unit documentation in PDF-format available:

- Safety instructions.
- Ethernet & Sequencer programming manual.

4.3.5 Ethernet

The ETH interface is available 15s after start up of the unit.

Connect the unit to the network via the LAN-connector at the rear.

Set the programming source for voltage and/or current to 'eth' via the eth commands or the web interface.

4.3.6 Sequencer

Define a sequence using a basic text editor, for example Notepad. Save as "filename.seq". An example is shown in chapter 7 of this manual. Upload the sequence via the web interface or via Eth programming commands.

Set the programming source for voltage and/or current to 'seq' via the web interface or Eth commands.

Start/Stop the sequence via the web interface, Eth commands or a hardware trigger via the Digital I/O interface.

Note: copy the uploaded sequences into the non-volatile memory before switching off the unit. Standard they are uploaded in the volatile memory and are lost after switching off the mains.

4.4 Forgotten password, access key or network settings

To reset the password and the network settings to their default values, press the reset button at the rear panel of the unit (while the unit is switched on).

A bent paperclip can be used to press the internal micro push button. A soft sensible click can be noticed. Press and hold the button for at least 4 seconds in order to activate the default restoring mechanism.

5 Conventions

The PSC-ETH-2 has a command set, which includes commands compatible with the SCPI language (Standard Commands for Programmable Instruments).

5.1 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-2.

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).

5.2 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.

5.3 Space <sp>

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

5.4 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>.

Default terminator is a linefeed (ASCII character OAH, 10d).

For other terminator options see paragraph 6.6.8, sub-paragraph "Terminator".

5.5 Parameters

Within this document, parameters are used to indicate the form of data sent to or coming from the PSC-ETH-2.

<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.

6 Command description

6.1 General Instructions

6.1.1 *IDN?

This command is used to read the identification string of the PSC-ETH-2. The string contains the name, the option number, the version of the firmware and the serial number of the PSC.

Syntax: ***IDN?<term>**

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<term>
```

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-2 interface.

The last field is reserved for future implementations.

6.1.2 *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 14 2019".

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>**

6.1.3 *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<term>**

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

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

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

6.1.4 *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-2:

Setting Value set after *RST: set after power-on:

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

Table 6.1

6.1.5 *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<term>**

6.2 Source Subsystem

6.2.1 Introduction

First of all the PSC-ETH-2 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 / 5 A. These settings must be changed for correct operation. When the PSC-ETH-2 is built-in by Delta Elektronika B.V., 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.

6.2.2 Maximum Output Voltage

To set the maximum output voltage of the power supply, this command must be sent. Refer to section 6.1.3 how to save this setting to the non-volatile memory.

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

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

To read the last settings, send the query:

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

For example the answer is: 30.0000<term>

6.2.3 Maximum Output Current

To set the maximum output current of the power supply, this command must be sent. Refer to section 6.1.3 how to save this setting to the non-volatile memory.

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

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

To read the last settings, send the query:

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

6.2.4 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>**

6.2.5 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>**

6.2.6 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.

6.2.7 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.

6.3 Measure Subsystem

6.3.1 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 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.

6.3.2 Measure Output Voltage

To measure the output voltage of the power supply:

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

6.3.3 Measure Output Current

To measure the output current of the power supply:

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

6.3.4 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?<term>**

6.4 Calibrate Subsystem

6.4.1 Introduction

To make the accuracy of the PSC-ETH-2 / power supply combination as high as possible, it is required to calibrate the PSC-ETH-2 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-2 is built-in by Delta Elektronika B.V.

However, periodical check and calibration is recommended.

At power-on 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<term>
- Calibrate the source voltage offset, so the output voltage is as close as possible to 0.6 V
- Calibrate the measure voltage offset, so the result of the command MEASure:VOLTage? is as close as possible to 0.6 V.
- Set the output voltage to maximum
- Calibrate the source voltage gain, so the output voltage is as close as possible to the maximum voltage, 60 V.
- Calibrate the measure voltage gain, so the result of the command MEASure:VOLTage? is as close as possible to the actual output voltage, 60 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 vice versa. To get a very accurate result, redo the calibration a few times.

There are 2 sets of commands to perform the calibration.

Set A is compatible with the first version of the PSC-ETH card.

Set B is compatible with the modern commands that are used in the latest SM-series.

Note that both sets have their own equations for the offset calculation.

Both sets of commands, A and B, can be used in a mixed way. There is no strict need or recommendation to choose for a certain set of commands.

6.4.2 Definitions

6.4.2.1 Programmed Value

The setting applied via web or SOURce:VOLTage or SOURce:CURRent.

6.4.2.2 Actual Value

The real output voltage or current measured with external instruments.

6.4.2.3 Measured Value

The response of MEASure:VOLTage? or MEASure:CURRent?.

6.4.3 Set A: compatible with PSC ETH

For SOURCE offset calibration, use the equation:

$$\text{new}_{\text{offset}} = \text{old}_{\text{offset}} + \left(\frac{\text{programmedValue} - \text{actualValue}}{\text{MaximumOutput}} \right) * 5$$

For SOURCE gain calibration, use the equation:

$$\text{new}_{\text{gain}} = \text{old}_{\text{gain}} * \left(\frac{\text{programmedValue}}{\text{actualValue}} \right)$$

For MEASURE offset calibration, use the equation:

$$\text{new}_{\text{offset}} = \text{old}_{\text{offset}} + \left(\frac{\text{actualValue} - \text{measuredValue}}{\text{MaximumOutput}} \right) * 5$$

For MEASURE gain calibration, use the equation:

$$\text{new}_{\text{gain}} = \text{old}_{\text{gain}} * \left(\frac{\text{actualValue}}{\text{measuredValue}} \right)$$

To save the calibration settings to the non-volatile memory, refer to section 6.1.

6.4.3.1 Calibrate Gain Source Current

To calibrate the programming gain of the current setting:

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

To read the calibration settings:

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

6.4.3.2 Calibrate Offset Source Current

To calibrate the programming offset of the current setting:

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

To read the calibration settings:

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

6.4.3.3 Calibrate Gain Source Voltage

To calibrate the programming gain of the voltage setting:

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

To read the calibration settings:

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

6.4.3.4 Calibrate Offset Source Voltage

To calibrate the programming offset of the voltage setting:

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

To read the calibration settings:

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

6.4.3.5 Calibrate Gain Measure Current

To calibrate the gain of the current measurement:

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

To read the calibration settings:

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

6.4.3.6 Calibrate Gain Measure Voltage

To calibrate the gain of the voltage measurement:

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

To read the calibration settings:

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

6.4.3.7 Calibrate Offset Measure Current

To calibrate the offset of the current measurement:

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

To read the calibration settings:

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

6.4.3.8 Calibrate Offset Measure Voltage

To calibrate the offset of the voltage measurement:

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

To read the calibration settings:

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

6.4.4 Set B: compatible with PSC-ETH-2 / SM-Series

For SOURCE offset calibration, use the equation:

$$\text{new}_{\text{offset}} = \text{old}_{\text{offset}} + (\text{programmedValue} - \text{actualValue})$$

For SOURCE gain calibration, use the equation:

$$\text{new}_{\text{gain}} = \text{old}_{\text{gain}} * \left(\frac{\text{programmedValue}}{\text{actualValue}} \right)$$

For MEASURE offset calibration, use the equation:

$$\text{new}_{\text{offset}} = \text{old}_{\text{offset}} + (\text{actualValue} - \text{measuredValue})$$

For MEASURE gain calibration, use the equation:

$$\text{new}_{\text{gain}} = \text{old}_{\text{gain}} * \left(\frac{\text{actualValue}}{\text{measuredValue}} \right)$$

To save the calibration settings to the non-volatile memory, refer to section 6.1.

6.4.4.1 Calibrate Gain Source Current

To calibrate the gain of the current setting:

Syntax: **CALibrate:CURrent:GAIn<sp><NR2><term>**

To read the calibration settings:

Syntax: **CALibrate:CURrent:GAIn?<term>**

6.4.4.2 Calibrate Gain Source Voltage

To calibrate the gain of the voltage setting:

Syntax: **CALibrate:VOLtage:GAIn<sp><NR2><term>**

To read the calibration settings:

Syntax: **CALibrate:VOLtage:GAIn?<term>**

6.4.4.3 Calibrate Offset Source Current

To calibrate the offset of the current setting:

Syntax: **CALibrate:CURrent:OFFset<sp><NR2><term>**

To read the calibration settings:

Syntax: **CALibrate:CURrent:OFFset?<term>**

6.4.4.4 Calibrate Offset Source Voltage

To calibrate the offset of the voltage setting:

Syntax: **CALibrate:VOLtage:OFFset<sp><NR2><term>**

To read the calibration settings:

Syntax: **CALibrate:VOLtage:OFFset?<term>**

6.4.4.5 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>**

6.4.4.6 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>**

6.4.4.7 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>**

6.4.4.8 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>**

6.5 Digital User In-/Outputs

The SM800, SM1500, SM6000 and the external PSC-ETH-2 provide 8 user inputs and 8 outputs. These can be controlled / monitored by commands (explained below) or can be used to interact with the Sequencer (refer to chapter 7).

6.5.1 User Outputs

To program the user outputs, a decimal number can be send to the PSC-ETH-2. This decimal number represents the binary state of the 8 user outputs.

Syntax: **UOUTput<sp><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 F, send **UOUTput<sp><36><term>**. (=4+32)

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

To read the last setting of the user outputs:

Syntax: **UOUTput?<term>**

6.5.2 User Inputs

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

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

The PSC-ETH-2 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)

6.6 System Subsystem

6.6.1 Remote Shut Down (RSD)

Syntax: **SYSTem:RSD[:STATus]<sp><boolean><term>**

boolean = 0, 1, OFF, ON

To read the last setting:
 Syntax: **SYSTem:RSD[:STATus]?<term>**

6.6.2 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?**

6.6.3 Front Panel Lock

This command is only available for PSC-ETH-2, which are built inside a power supply with internal CPU; SM800, SM1500 and SM6000.
 Syntax: **SYSTem:FRONTpanel[:STATus]<sp><boolean><term>** boolean = 0, 1, OFF, ON
 To read the last setting:
 Syntax: **SYSTem:FRONTpanel[:STATus]?<term>**

6.6.4 Remote programming

The built-in PSC-ETH-2 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 below table 5.2.

6.6.4.1 Remote method 1

With this method the two programming signals CV and CC are switched simultaneously.
 Syntax: **SYSTem:REMote[:STATus]<sp><setting><term>** setting = REMote or LOCal¹
 To read the last setting:
 Syntax: **SYSTem:REMote[:STATus]?<term>**

6.6.4.2 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-2 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><term>** setting = REMote or LOCal¹
 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 LOCal¹
 To read the last setting:
 Syntax: **SYSTem:REMote:CC[:STATus]?<term>**

1) The settings Remote or Local are equal to the settings Ethernet or Front.

Possible settings are: Front (Local), Web, Sequencer or Ethernet (Remote).

Type of unit	REMOTE	REMOTE:CV	REMOTE:CC
External	no	no	no
ES150	no	no	no
ES300	no	no	no
SM800	no	yes	yes
SM1500	no	yes	yes
SM6000	no	yes	yes
SM1500-1 * Obsolete (no longer manufactured)	yes	no	no
SM3000 Obsolete (no longer manufactured)	no	no	no

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

Table 6.2

6.6.5 Error Message

If an unknown command, an invalid value or an illegal setting is received by the PSC-ETH-2 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 PSC-ETH-2 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.

6.6.6 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:PAStword<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. The Network settings will also be set to factory default "DHCP Enable". In the beginning of this manual there is a detailed description about the reset switch.

To store the new password, refer to section 6.1.3 (*SAV)

To read if a password is used, send the query

Syntax : **SYSTem:PAStword:STAtus?<term>**

If no password is used, the PSC-ETH-2 will return 0<term>, otherwise it returns 1<term>.

6.6.7 Watchdog

The PSC-ETH-2 provides a Watchdog timer on the Ethernet interface. The PSC-ETH-2 monitors the Ethernet communication when set and disables its power output when no Ethernet command is received within the time set.

This function is only supported for power supplies, which have a push button "OUTPUT ON" on the front panel and with the PSC-ETH-2 build-in.

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 the Web
-1<term>	Clears Timeout, Watchdog is off

To disable the Watchdog timer:

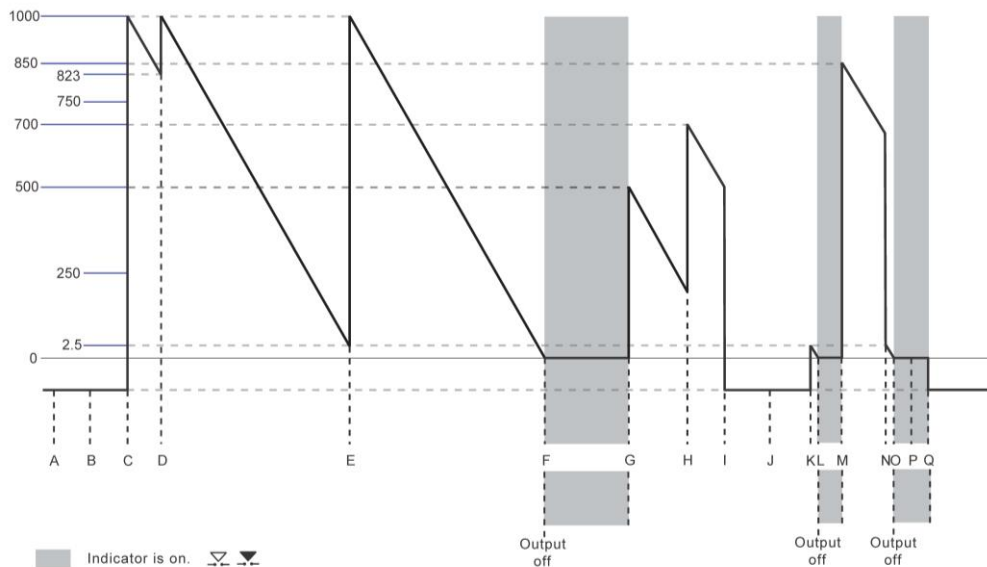
Syntax : **SYSTem:COMMunicate:WATChdog<sp>STOP<term>**

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 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 webserver.

Syntax : **SYSTEM:COMMunicate:WATchdog?<term>**

Q = Gives -1, Indicates that the watchdog is still off and makes the count -1.

6.6.8 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>**

6.7 Output

The PSC-ETH-2 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 and with the PSC-ETH-2 build-in.

Refer to the manual of the power supply to learn more about this function.

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

To read the last settings:

Syntax: **OUTPut?<term>**

6.8 Register Structure

The PSC-ETH-2 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 PSC-ETH-2 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	Reserved
3	8	V _{lim} *
4	16	I _{lim} *
5	32	Reserved
6	64	DCF
7	128	Reserved
8	256	OT
9	512	PSOL
10	1024	ACF
11	2048	Reserved
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	Reserved
3	8	Program running
4	16	Wait for Trigger
5	32	Reserved
6	64	Reserved
7	128	V _{output} overload
8	256	I _{output} Overload
9	512	Reserved
10	1024	Reserved
11	2048	Reserved
12	4096	Reserved
13	8192	Reserved
14	16384	Reserved
15	32768	Program Open End Error (cleared after read)

* only available for PSC-ETH-2 in SM800, SM1500 or SM6000.

6.8.1 *CLS

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

Syntax: ***CLS<term>**

7 Sequencer

7.1 Introduction

The PSC-ETH-2 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 7.3) or by a user input (see section 7.4). 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.

7.2 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.

7.2.1 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 H (output A to H). This command can set or reset an digital output.

Syntax: **Ox=<boolean>** <boolean> = 0 or 1 x = A,B,C,D,E,F,G or H

#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

7.2.2 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 6 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,F,G or H
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,F,G or H
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 CJG<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. 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 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><SC>,<NR2>,<step>
CJL<sp><MC>,<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.

7.2.3 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

7.2.4 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

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

7.3 Sequence control by commands

7.3.1 Read the Catalog

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

Syntax : **PROG:CATalog?<term>**

The PSC-ETH-2 returns a list of the names of all sequences, separated by linefeeds.

The end of the catalog is indicated by an extra <term>.

In case of no sequences available, the PSC-ETH-2 only returns one <term>.

Example: *PROG:CAT?<term>* answer: *WAVE1<nl>PROCESS4<nl>RAMPUP<nl><term>*

7.3.2 How to create or select a Sequence

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

Syntax: **PROG: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: **PROG:SElected:NAME?<term>**

The PSC-ETH-2 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.

7.3.3 Upload a Sequence to PSC-ETH-2 (PC → PS)

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

Syntax: **PROG: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>*

7.3.4 Download a Sequence from PSC-ETH-2 (PS → PC)

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

Syntax: **PROG:SElected:STEp<sp><NR1>?<term>**

The PSC-ETH-2 returns <step number><sp><command+operand(s)><term>.

If the queried step doesn't exist, the PSC-ETH-2 returns <term>.

The PSC-ETH-2 returns the complete sequence when it receives the query:

Syntax: **PROG:SElected:STEp<sp>?<term>**

The answer from the PSC-ETH-2 will be:

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

After the last step the PSC-ETH-2 sends <term> as a terminator.

7.3.5 Delete a Sequence

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

Syntax: **PROG:SElected:DElete<term>**

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:

Syntax: **PROG:CATalog:DElete<term>**

7.3.6 Start a Sequence

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

Syntax: **PROG: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.

7.3.7 Pause a Sequence

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

Syntax: **PROG:SElected:STate<sp>PAUSE<term>**

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

Syntax: **PROG:SElected:STate<sp>CONTINUE<term>**

7.3.8 Step through a Sequence

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

Syntax: **PROG: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 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-2 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.

For example:

```
.....
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.

7.3.9 Stop a Sequence

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

Syntax: **PROG:SElected:STAtE<sp><STOP><term>**

The sequence will stop immediately, but is still selected.

7.3.10 Read Sequence mode

By sending the query:

Syntax: **PROG:SElected:STAtE?<term>**

The PSC-ETH-2 will return the current mode. There are three possibilities:

STOP<term>

PAUSE,<next step><term>

RUN,<next step><term>

Syntax: **PROG:SElected:STAtE<SP>active?<term>**

The PSC-ETH-2 will return the current mode.

There are three possibilities:

STOP<term>

PAUSE,<active step><term>

RUN,<active step><term>

7.3.11 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.

7.3.12 Add labels

Use the following command to define labels:

Syntax: **PROG:SElected:LABel<sp><name>,<step><term>** name = Label name

To query the active Labels, use the command:

Syntax: **PROG: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 7.4.13 for example.

7.3.13 Delete labels

Delete a Label by sending the command:

Syntax: **PROG:SElected:LABel<sp><NAME>,DELETE<term>**

Use the following command to delete all Labels:

Syntax: **PROG:SElected:LABel<sp><*>,DELETE<term>**

7.3.14 Building a Sequence

Before a sequence can be started it has to be build. During a build the PSC-ETH-2 will, for example, check if all used Labels are defined properly.

7.4.2 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 w=1
4 #j=3
5 inc sv,5
20 cjne #j,0,5
21 sv=100
...
    
```

7.4.3 How to select a sequence

Sequences available on 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.

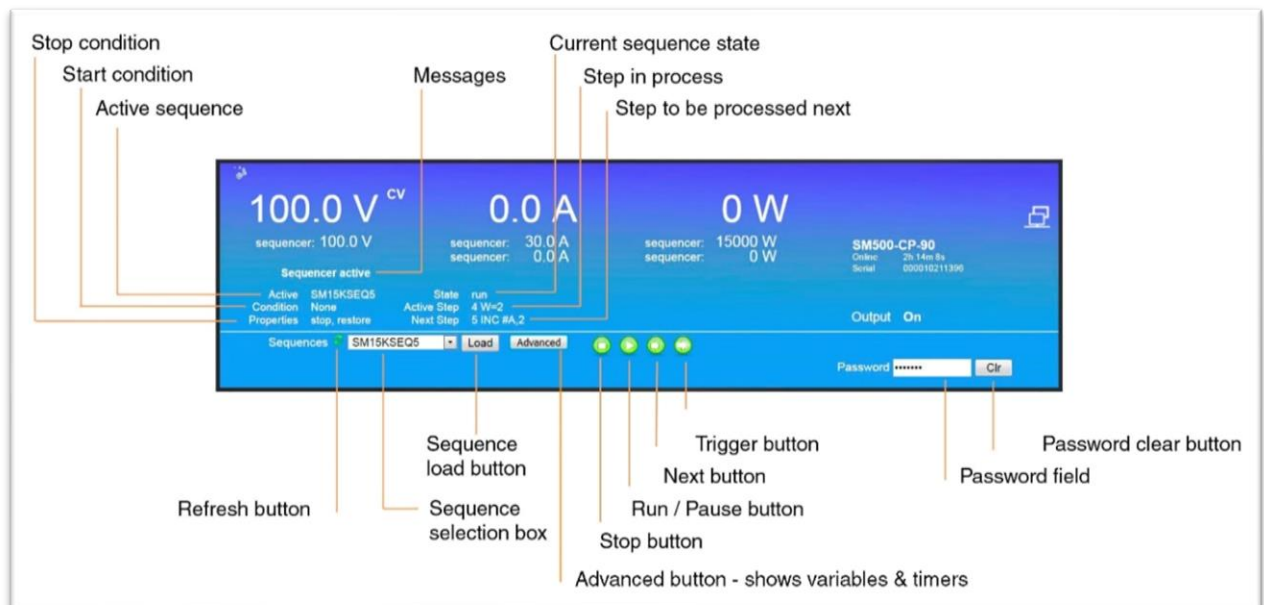


Fig 7.2 - Sequencer console

7.4.4 Upload a sequence to PSC-ETH-2 (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 7.3.14 Building a Sequence) Click on Back to find the sequence in the catalog when the upload is accepted by the power supply.

7.4.5 Download a Sequence from PSC-ETH-2 (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.

7.4.6 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 the Web 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 15 seconds in which the web will wait.

7.4.7 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.

7.4.8 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.

7.4.9 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.

7.4.10 Stop a Sequence

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

7.4.11 Read Sequence mode

State <current state> on the Sequencer Console displays the current state. There are three possibilities: Stop, Pause and Run.

7.4.12 Trigger a Step

When a sequence contains TRG, the sequence waits for the command via TCP/IP (section 7.3.11 Trigger a step). Triggering a step by the Web Console is not implemented.

7.4.13 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.

7.4.14 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.

7.4.15 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 7.3 (Sequence control by commands)

7.4.16 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 Web password when required. Click on Sync memory to update the non-volatile memory with the new setting. Updating the non-volatile memory takes about 15 seconds in which the web will wait.

7.5 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-2 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.

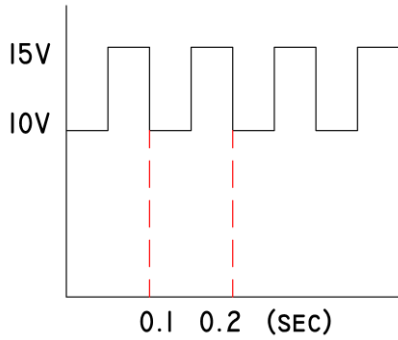
Any combination of the user inputs and the options can be made. It is required to set every character of the assignment. Refer to section to find the command to program the sequence name. The length of the sequence name + the assignment has a maximum of 16 characters.

7.6 Sequence examples

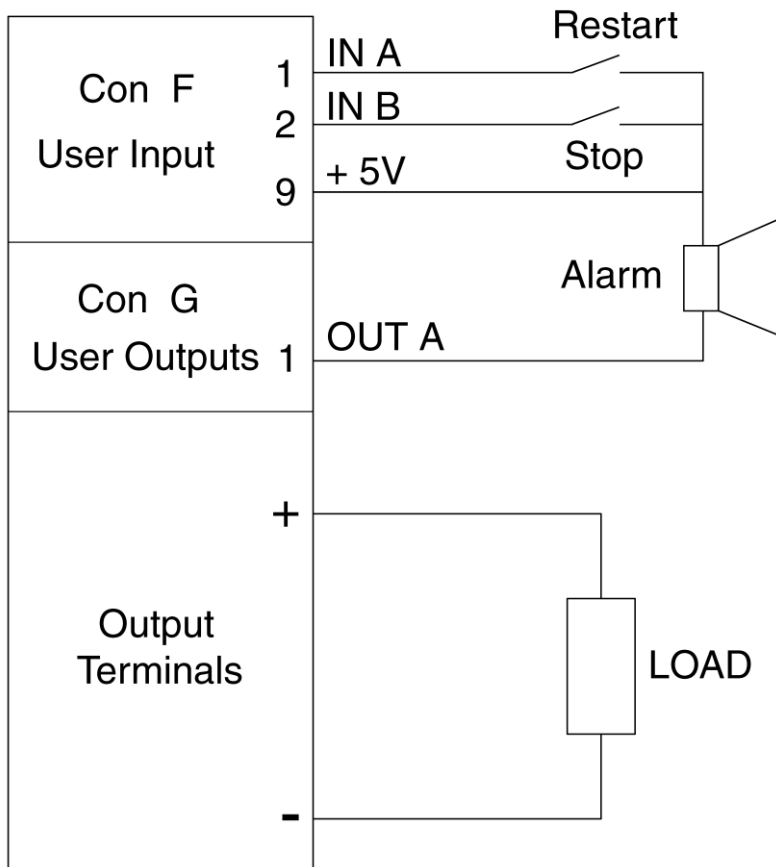
In this paragraph a few examples are explained based on typical applications. Every example contains a short description of the application, a flowchart, the list of the required sequence steps and the commands how to upload the sequence into the PSC-ETH-2.

7.6.1 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

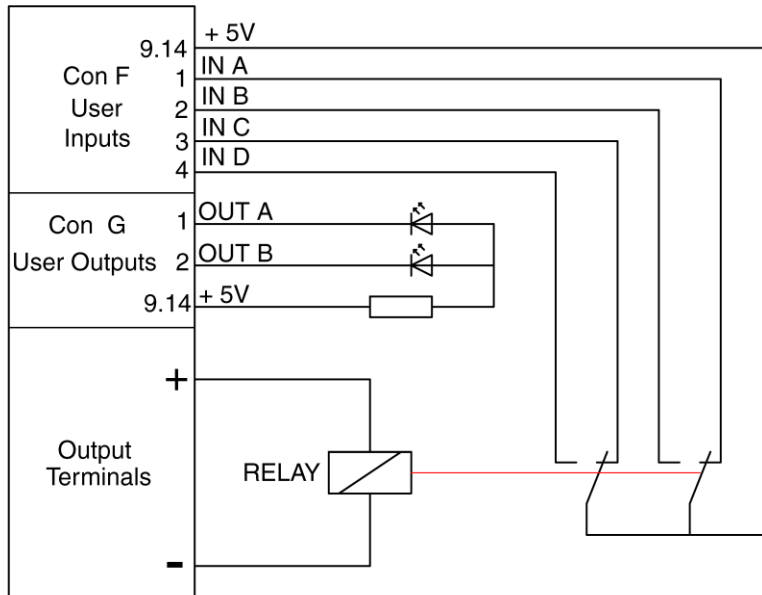
```


7.6.2 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 oa=0
2 ob=0
3 js 21
4 nop
5 w=1
6 sv=5.9
7 cjne ia,1,30
8 cjne ib,0,30
9 cjne ic,1,30
10 cjne id,0,30
11 cjb sv,11.8,30
12 inc sv,0.05
13 w=0.1
14 cjne ia,1,34
15 cjne ib,0,34
16 cjne ic,1,34
17 cjne id,0,34
18 jp 11
19 end
20 nop
21 sv=5
22 sc=0.3
23 w=0.1
24 cjb mc,0.01,29
25 oa=1
26 ob=1
27 w=1
28 jp 19
29 ret
30 oa=1
31 w=1
32 jp 19
33 nop
34 ob=1
35 w=1
36 jp 19
37 nop

```

8 Command list TCP/IP

8.1 Index TCP / IP

*IDN?<term>	13
*PUD<sp><data><term>	13
*PUD?<term>	13
*SAV<term>	13
*SAV<sp><password><term>	13
*RCL<term>	14
SOURce:VOLTage:MAXimum<sp><NR2><term>	14
SOURce:VOLTage:MAXimum?<term>	14
SOURce:CURRent:MAXimum<sp><NR2><term>	14
SOURce:CURRent:MAXimum?<term>	14
SOURce:VOLTage<sp><NR2><term>	14
SOURce:VOLTage?<term>	14
SOURce:CURRent<sp><NR2><term>	14
SOURce:CURRent?<term>	14
SOURce:VOLTage:STEPSize?<term>	14
SOURce:CURRent:STEPSize?<term>	14
MEASure:VOLTage?<term>	15
MEASure:CURRent?<term>	15
MEASure:POWER?<term>	15
CAL<sp>0,<NR2><term>	16
CAL<sp>0?<term>	16
CAL<sp>1,<NR2><term>	16
CAL<sp>1?<term>	16
CAL<sp>2,<NR2><term>	16
CAL<sp>2?<term>	16
CAL<sp>3,<NR2><term>	16
CAL<sp>3?<term>	16
CAL<sp>4,<NR2><term>	16
CAL<sp>4?<term>	16
CAL<sp>5,<NR2><term>	17
CAL<sp>5?<term>	17
CAL<sp>6,<NR2><term>	17
CAL<sp>6?<term>	17
CAL<sp>7,<NR2><term>	17
CAL<sp>7?<term>	17
CALlbrate:CURRent:GAIN<sp><NR2><term>	17
CALlbrate:CURRent:GAIN?<term>	17
CALlbrate:VOLTage:GAIN<sp><NR2><term>	17
CALlbrate:VOLTage:GAIN?<term>	17
CALlbrate:CURRent:OFFset<sp><NR2><term>	17
CALlbrate:CURRent:OFFset?<term>	17
CALlbrate:VOLTage:OFFset<sp><NR2><term>	17
CALlbrate:VOLTage:OFFset?<term>	17
CALlbrate:CURRent:MEASure:GAIN<sp><NR2><term>	18
CALlbrate:CURRent:MEASure:GAIN?<term>	18
CALlbrate:VOLTage:MEASure:GAIN<sp><NR2><term>	18

CALibrate:VOLtage:MEASURE:GAIN?<term> 18

CALibrate:CURrent:MEASURE:OFFset<sp><NR2><term> 18

CALibrate:CURrent:MEASURE:OFFset?<term> 18

CALibrate:VOLtage:MEASURE:OFFset<sp><NR2><term> 18

CALibrate:VOLtage:MEASURE:OFFset?<term> 18

UOUTput<sp><0+NR1><term> 18

UOUTput?<term> 18

UINPut:CONDition?<term> 18

SYSTem:RSD[:STATus]<sp><boolean><term> 18

SYSTem:RSD[:STATus]?<term> 19

SYSTem:LIMits:VOLtage<sp><NR2>,<boolean><term> 19

SYSTem:LIMits:VOLtage? 19

SYSTem:LIMits:CURrent<sp><NR2>,<boolean><term> 19

SYSTem:LIMits:CURrent? 19

SYSTem:FRONtpanel[:STATus]<sp><boolean><term> 19

SYSTem:FRONtpanel[:STATus]?<term> 19

SYSTem:REMote[:STATus]<sp><setting><term> 19

SYSTem:REMote[:STATus]?<term> 19

SYSTem:REMote:CV[:STATus]<sp><setting><term> 19

SYSTem:REMote:CV[:STATus]?<term> 19

SYSTem:REMote:CC[:STATus]<sp><setting><term> 19

SYSTem:REMote:CC[:STATus]?<term> 19

SYSTem:ERRor?<term> 20

SYSTem:PASsword<sp><old_password>,<new_password><term> 20

SYSTem:PASsword:STATus?<term> 20

SYSTem:COMmunicate:WATchdog<sp>SET,<NR1><term> 20

SYSTem:COMmunicate:WATchdog<sp>SET?<term> 20

SYSTem:COMmunicate:WATchdog?<term> 20

SYSTem:COMmunicate:WATchdog<sp>STOP<term> 20

SYSTem:COMmunicate:WATchdog<sp>TEST<term> 20

SYSTem:COMmunicate:TERminator<sp> <value> <term> 21

SYSTem:COMmunicate:TERminator?<term> 22

OUTPut<sp><boolean><term> 22

OUTPut?<term> 22

STATus:REGister:A?<term> 22

STATus:REGister:B?<term> 22

***CLS<term> 22**

9 Command list Sequencer

9.1 Index Sequencer

SV=<NR2>	23
SC=<NR2>.....	23
Ox=<boolean>	23
#x=<NR1>.....	23
#l=<NR1>.....	23
#J=<NR1>.....	23
JP<sp><step>.....	23
JS<sp><step>.....	23
RET	23
CJE<sp><lx>,<boolean>,<step>.....	24
CJE<sp><Ox>,<boolean>,<step>	24
CJE<sp><#x>,<NR1>,<step>	24
CJNE<sp><lx>,<boolean>,<step>	24
CJNE<sp><Ox>,<boolean>,<step>	24
CJNE<sp><#x>,<NR2>,<step>.....	24
CJG<sp><SV>,<NR2>,<step>	24
CJG<sp><MV>,<NR2>,<step>.....	24
CJG<sp><SC>,<NR2>,<step>	24
CJG<sp><MC>,<NR2>,<step>	24
CJG<sp><#x>,<NR1>,<step>	24
CJL<sp><SV>,<NR2>,<step>.....	24
CJL<sp><MV>,<NR2>,<step>	24
CJL<sp><SC>,<NR2>,<step>.....	24
CJL<sp><MC>,<NR2>,<step>	24
CJL<sp><#x>,<NR1>,<step>	24
INC<sp><SV>,<NR2>	24
INC<sp><SC>,<NR2>	24
INC<sp><#x>,<NR1>	24
DEC<sp><SV>,<NR2>.....	24
DEC<sp><SC>,<NR2>	24
DEC<sp><#x>,<NR1>	24
NOP	25
W=<NR2>	25
TRG.....	25
END.....	25
PROGram:CATalog?<term>.....	26
PROGram:SElected:NAME<sp><string><term>	26
PROGram:SElected:NAME?<term>	26
PROGram:SElected:STEp<sp><NR1><sp><command+operand(s)><term>	26
PROGram:SElected:STEp<sp><NR1>?<term>.....	26
PROGram:SElected:STEp<sp>?<term>	26
PROGram:SElected:DELeTe<term>	26
PROGram:SElected:STAtE<sp>RUN<term>	26
PROGram:SElected:STAtE<sp>PAUSE<term>	26
PROGram:SElected:STAtE<sp>CONTInue<term>	26
PROGram:SElected:STAtE<sp>NEXT<term>	26

PROG ram:SELEcted:STAtE<sp><STOP><term>	27
PROG ram:SELEcted:STAtE?<term>	27
PROG ram:SELEcted:STAtE<SP>active?<term>	27
TRIG ger:IMMEdiate<term>	27
PROG ram:SELEcted:LABel<sp><name>,<step><term>	27
PROG ram:SELEcted:LABel<sp>?<term>	27
PROG ram:SELEcted:LABel<sp><NAME>,<DELETE><term>	27
PROG ram:SELEcted:LABel<sp><*>,<DELETE><term>	27
PROG ram:SELEcted:BUId<term>	28
PROG ram:SELEcted:BUId?<term>	28
PROG ram:SELEcted:NONvolatile<sp><boolean><term>	28
PROG ram:SELEcted:NONvolatile?<term>	28
PROG ram:SAVe<term>	28
PROG ram:SAVe?<term>	28



10 EU-Declaration of Conformity - PSC-ETH Gen2



We

Delta Elektronika
Vissersdijk 4
4301 ND ZIERIKZEE
The Netherlands

Declare under sole responsibility that the following Interfaces:

PSC-ETH-2 CARD
PSC-ETH-2 MODULE

Meet the intent of Directives

2014/30/EU Electromagnetic Compatibility (EMC)
2014/35/EU Low Voltage Directive (LVD)
2011/65/EU Reduction of Hazardous Substances (RoHS2)

Compliance was demonstrated to the following specification as listed in the official Journal of the European Union:

EN 61326-1:2013

EMC requirements for electrical equipment for measurement, control and laboratory use

EN 61010-1:2010/A1:2019/AC:2019-04

Safety requirements for electrical equipment for measurement, control and laboratory use

EN IEC 63000:2018

Assessment of electrical and electronic products with respect to RoHS

J. Koopman
Managing director,
Zierikzee, March 2025