



EEZ Programmable PSU SCPI reference guide



Firmware version: M2

Platform: EEZ PSU with Arduino Shield +BP r1B9 or latter

Document name: EEZ PSU SCPI reference for M2

Date: 2016-06-03



This document is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Table of Contents

| | |
|---|----|
| 1.Introduction..... | 7 |
| 1.1.About SCPI..... | 7 |
| 2.Syntax and style..... | 9 |
| 2.1.Root Specifier..... | 9 |
| 2.2.Command termination..... | 9 |
| 2.3.Command separators..... | 9 |
| 2.4.Querying parameter settings..... | 9 |
| 2.5.Using the MIN, MAX, and DEF Parameters..... | 9 |
| 2.6.Command and message types..... | 10 |
| 2.7.Required Commands..... | 10 |
| 2.7.1.Base functionality for the Power supply instrument class..... | 10 |
| 2.8.Multiple Commands in a Message..... | 11 |
| 2.9.Moving Among Subsystems..... | 11 |
| 2.10.SCPI parameter types..... | 12 |
| 3.Registers and queues..... | 13 |
| 3.1.Standard Event Status Register..... | 14 |
| 3.2.Status Byte Register..... | 15 |
| 3.3.OPERation Status Register..... | 16 |
| 3.3.1.Operation INSTRument Status register..... | 17 |
| 3.3.2.Operation Instrument SUMmary status register..... | 17 |
| 3.4.QUESTionable Status Register..... | 17 |
| 3.4.1.Questionable INSTRument Status register..... | 19 |
| 3.4.2.Questionable Instrument SUMmary status register..... | 19 |
| 3.5.Error queue..... | 20 |
| 4.Common command reference..... | 21 |
| 4.1.*CLS..... | 21 |
| 4.2.*ESE..... | 21 |
| 4.3.*ESR?..... | 22 |
| 4.4.*IDN?..... | 22 |
| 4.5.*OPC..... | 22 |
| 4.6.*RCL..... | 23 |
| 4.7.*RST..... | 23 |
| 4.8.*SAV..... | 24 |
| 4.9.*STB?..... | 24 |
| 4.10.*TST?..... | 25 |
| 5.Subsystem command reference..... | 27 |
| 5.1.CALibrate..... | 27 |
| 5.1.1.CALibration[:MODE]..... | 27 |
| 5.1.2.CALibration:CLEar..... | 28 |
| 5.1.3.CALibration:CURREnt[:DATA]..... | 28 |
| 5.1.4.CALibration:CURREnt:LEVel..... | 29 |
| 5.1.5.CALibration:PASSword:NEW..... | 29 |
| 5.1.6.CALibration:REMark..... | 29 |
| 5.1.7.CALibration:SAVE..... | 30 |
| 5.1.8.CALibration:STATe..... | 30 |
| 5.1.9.CALibration:VOLTage[:DATA]..... | 30 |
| 5.1.10.CALibration:VOLTage:LEVel..... | 31 |
| 5.2.DIAGnostic..... | 33 |
| 5.2.1.DIAGnostic[:INFOrmation]:ADC?..... | 33 |
| 5.2.2.DIAGnostic[:INFOrmation]:FAN?..... | 33 |

| | |
|--|----|
| 5.2.3.DIAGnostic[:INFOrmation]:CALibration? | 33 |
| 5.2.4.DIAGnostic[:INFOrmation]:PROTection? | 34 |
| 5.2.5.DIAGnostic[:INFOrmation]:TEST? | 35 |
| 5.3.INSTrument | 37 |
| 5.3.1.INSTrument[:SElect] | 37 |
| 5.3.2.INSTrument:NSElect | 37 |
| 5.4.MEASure | 39 |
| 5.4.1.MEASure[:SCALar]:CURRent[:DC] | 39 |
| 5.4.2.MEASure[:SCALar]:POWEr[:DC] | 39 |
| 5.4.3.MEASure[:SCALar]:TEMPerature[:THERmistor][:DC] | 39 |
| 5.4.4.MEASure[:SCALar][:VOLTage][:DC] | 40 |
| 5.5.MEMory | 41 |
| 5.5.1.MEMory:NStates | 41 |
| 5.5.2.MEMory:STATe:CATalog | 41 |
| 5.5.3.MEMory:STATe:DELEte | 41 |
| 5.5.4.MEMory:STATe:NAME | 42 |
| 5.5.5.MEMory:STATe:RECall:AUTO | 42 |
| 5.5.6.MEMory:STATe:RECall:SElect | 43 |
| 5.5.7.MEMory:STATe:VALid | 43 |
| 5.6.OUTPut | 45 |
| 5.6.1.OUTPut[:STATe] | 45 |
| 5.6.2.OUTPut:DPRog | 46 |
| 5.6.3.OUTPut:MODE? | 46 |
| 5.6.4.OUTPut:PROTection:CLEAr | 47 |
| 5.7.SOURce | 49 |
| 5.7.1.[SOURce[<n>]]:CURRent | 50 |
| 5.7.2.[SOURce[<n>]]:CURRent:STEP | 50 |
| 5.7.3.[SOURce[<n>]]:CURRent:PROTection:DELAy[:TIME] | 51 |
| 5.7.4.[SOURce[<n>]]:CURRent:PROTection:STATe | 52 |
| 5.7.5.[SOURce[<n>]]:CURRent:PROTection:TRIPped? | 52 |
| 5.7.6.[SOURce[<n>]]:LRIPple | 52 |
| 5.7.7.[SOURce[<n>]]:LRIPple:AUTO | 53 |
| 5.7.8.[SOURce[<n>]]:POWEr:PROTection[:LEVel] | 54 |
| 5.7.9.[SOURce[<n>]]:POWEr:PROTection:DELAy[:TIME] | 54 |
| 5.7.10.[SOURce[<n>]]:POWEr:PROTection:STATe | 55 |
| 5.7.11.[SOURce[<n>]]:POWEr:PROTection:TRIPped? | 55 |
| 5.7.12.[SOURce[<n>]]:VOLTage | 55 |
| 5.7.13.[SOURce[<n>]]:VOLTage:STEP | 56 |
| 5.7.14.[SOURce[<n>]]:VOLTage:PROGram[:SOURce] | 57 |
| 5.7.15.[SOURce[<n>]]:VOLTage:PROTection:DELAy[:TIME] | 58 |
| 5.7.16.[SOURce[<n>]]:VOLTage:PROTection:STATe | 58 |
| 5.7.17.[SOURce[<n>]]:VOLTage:PROTection:TRIPped? | 58 |
| 5.7.18.[SOURce[<n>]]:VOLTage:SENSE[:SOURce] | 59 |
| 5.8.STATus | 61 |
| 5.8.1.STATus:OPERation[:EVENT]? | 62 |
| 5.8.2.STATus:OPERation:CONDition? | 62 |
| 5.8.3.STATus:OPERation:ENABLE | 62 |
| 5.8.4.STATus:OPERation:INSTrument[:EVENT]? | 63 |
| 5.8.5.STATus:OPERation:INSTrument:CONDition? | 63 |
| 5.8.6.STATus:OPERation:INSTrument:ENABLE | 63 |
| 5.8.7.STATus:OPERation:INSTrument:ISUMmary[<n>][:EVENT]? | 64 |
| 5.8.8.STATus:OPERation:INSTrument:ISUMmary[<n>]:CONDition? | 64 |

| | |
|---|----|
| 5.8.9.STATUS:OPERation:INSTrument:ISUMmary<n>:ENABLE..... | 64 |
| 5.8.10.STATUS:PREset..... | 65 |
| 5.8.11.STATUS:QUEStionable[:EVENT]?..... | 65 |
| 5.8.12.STATUS:QUEStionable:CONDition?..... | 66 |
| 5.8.13.STATUS:QUEStionable:ENABLE..... | 66 |
| 5.8.14.STATUS:QUEStionable:INSTrument[:EVENT]?..... | 66 |
| 5.8.15.STATUS:QUEStionable:INSTrument:CONDition?..... | 67 |
| 5.8.16.STATUS:QUEStionable:INSTrument:ENABLE..... | 67 |
| 5.8.17.STATUS:QUEStionable:INSTrument:ISUMmary[<n>][:EVENT]?..... | 67 |
| 5.8.18.STATUS:QUEStionable:INSTrument:ISUMmary[<n>]:CONDition?..... | 68 |
| 5.8.19.STATUS:QUEStionable:INSTrument:ISUMmary[<n>]:ENABLE..... | 68 |
| 5.9.SYSTem..... | 71 |
| 5.9.1.SYSTem:BEEPer..... | 72 |
| 5.9.2.SYSTem:BEEPer:STATe..... | 72 |
| 5.9.3.SYSTem:CAPability?..... | 72 |
| 5.9.4.SYSTem:CHANnel[:COUNT]?..... | 72 |
| 5.9.5.SYSTem:CHANnel:INfOrmation:CURRent?..... | 72 |
| 5.9.6.SYSTem:CHANnel:INfOrmation:POWer?..... | 73 |
| 5.9.7.SYSTem:CHANnel:INfOrmation:PROGram?..... | 73 |
| 5.9.8.SYSTem:CHANnel:INfOrmation:VOLTage?..... | 73 |
| 5.9.9.SYSTem:CHANnel:MODEl?..... | 74 |
| 5.9.10.SYSTem:CPU:INfOrmation:ETHernet:TYPE?..... | 74 |
| 5.9.11.SYSTem:CPU:INfOrmation:TYPE?..... | 74 |
| 5.9.12.SYSTem:CPU:MODEl?..... | 74 |
| 5.9.13.SYSTem:CPU:OPTion?..... | 75 |
| 5.9.14.SYSTem:DATE..... | 75 |
| 5.9.15.SYSTem:ERRor..... | 75 |
| 5.9.16.SYSTem:ERRor:COUNT?..... | 76 |
| 5.9.17.SYSTem:POWer..... | 76 |
| 5.9.18.SYSTem:TEMPerature:PROTection[:HIGH][:LEVel]..... | 76 |
| 5.9.19.SYSTem:TEMPerature:PROTection[:HIGH]:CLEar..... | 77 |
| 5.9.20.SYSTem:TEMPerature:PROTection[:HIGH]:DELay[:TIME]..... | 77 |
| 5.9.21.SYSTem:TEMPerature:PROTection[:HIGH]:STATe..... | 78 |
| 5.9.22.SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped?..... | 78 |
| 5.9.23.SYSTem:TIME..... | 78 |
| 5.9.24.SYSTem:VERSion?..... | 79 |
| 6.Device-specific (unclassified) commands..... | 81 |
| 6.1.APPLY..... | 81 |
| 7.Error messages..... | 83 |
| 7.1.Command Error [-199, -100]..... | 83 |
| 7.2.Execution Error [-299, -200]..... | 84 |
| 7.3.Device-Specific Error [-399, -300], [1, 32767]..... | 85 |
| 7.3.1.Self-Test Error Messages..... | 86 |
| 8.Parameters and settings..... | 89 |
| 8.1.Programming parameters..... | 89 |
| 8.1.1.Voltage..... | 89 |
| 8.1.2.Current..... | 89 |
| 8.1.3.Power..... | 89 |
| 8.2.Reset Settings (*RST)..... | 90 |
| 9.Software simulator..... | 93 |
| 9.1.SIMUlator..... | 95 |
| 9.1.1.SIMUlator:EXIT..... | 95 |

| | |
|--|-----|
| 9.1.2.SIMUlator:GUI..... | 95 |
| 9.1.3.SIMUlator:LOAD..... | 95 |
| 9.1.4.SIMUlator:LOAD:STaTe..... | 96 |
| 9.1.5.SIMUlator:PWRGood..... | 96 |
| 9.1.6.SIMUlator:TEMPerature..... | 97 |
| 10.Programming examples..... | 99 |
| 10.1.Set channel output values and working with the OCP..... | 99 |
| 10.2.Voltage and current calibration..... | 101 |
| 10.3.Working with profiles..... | 101 |
| 10.4.Get identification info and self-test results..... | 103 |
| 11.SCPi commands scheduled for the Milestone Three (M3)..... | 105 |
| 12.SCPi commands summary..... | 109 |

1. Introduction

This manual contains reference information for programming open hardware/open source EEZ programmable PSU (Power Supply Unit) over the remote interface using the SCPI programming language. The SCPI (*Standard Commands for Programmable Instruments*, often pronounced “skippy”) is an open standard freely available on the [IVI Foundation](#) web pages. Current version is SCPI 1999.0. SCPI is pure software standard and SCPI syntax is ASCII text, and therefore can be attached to any computer language, such as C, C++, etc.

The physical communications link is not defined by SCPI. While originally created for IEEE 488 (GPIB), it can also be used with RS-232 (serial), Ethernet, USB, VXIbus, HiSLIP, etc. The EEZ PSU supports Serial (via USB) and Ethernet communication.

Application software that use SCPI commands is called *Controller* and SCPI enabled device such as PSU is called *Instrument*.

Please note that IEEE 488 standard documents are still not freely available, and if it is mentioned in this manual that is solely for reference purposes for one who wants to conduct a further research for better understanding or possible modification/improvement of the PSU remote control.

1.1. About SCPI

SCPI 1999.0 standard document says (Section 1.3) that the goal of SCPI is to reduce Automatic Test Equipment (ATE) program development time. SCPI accomplishes this goal by providing a consistent programming environment for instrument control and data usage. This consistent programming environment is achieved by the use of defined program messages, instrument responses, and data formats across all SCPI instruments, regardless of manufacturer.

A consistent program environment uses the same commands and parameters to control instruments that have the same functionality.

SCPI programming consistency is both vertical and horizontal. Vertical programming consistency defines program messages within an instrument class. An example of vertical consistency is using the same command for reading DC voltage from several different multimeters. Horizontal consistency is using the same command to control similar functions across instrument classes. For example, the trigger command would be the same for an identical trigger function found among counters, oscilloscopes, function generators, etc.

A key to consistent programming is the reduction of multiple ways to control similar instrument functions. The philosophy of SCPI is for the same instrument functions to be controlled by the same SCPI commands. To simplify learning, SCPI uses industry-standard names and terms that are manufacturer and customer supported.

SCPI is designed to be expanded with new defined commands in the future without causing programming problems. As new instruments are introduced, the intent is to maintain program compatibility with existing SCPI instruments.

Additional links:

- Wikipedia [SCPI](#)
- Technopedia [Standard Commands For Programmable Instruments \(SCPI\)](#)
- Wikipedia [IEEE-488](#)
- Keysight (Agilent) [Developing a SCPI command set](#)
- NI [GPIB Hardware and Software Specifications](#)

Implementation links:

- [Open source SCPI device](#) library
- Keysight (Agilent) [Application Note 1465-29](#)
- Keysight (Agilent) [Command Expert](#)

2. Syntax and style

Throughout this document, the following conventions are used for the SCPI command syntax:

- Square brackets ([]) indicate optional keywords or parameters. The braces are not sent with the command string.
- Braces ({}) enclose parameters within a command string.
- Triangle brackets (<>) indicate that you must substitute a value or a code for the enclosed parameter.
- A vertical bar (|) separates one of two or more alternative parameters.

2.1. Root Specifier

When it precedes the first header of a message unit, the colon becomes the root specifier. It tells the command parser that this is the root or the top node of the command tree.

2.2. Command termination

A command string sent to the PSU must terminate with a <new line>character. A <carriage return> followed by a <new line>is also accepted. Command string termination will always reset the current SCPI command path to the root level.

2.3. Command separators

A colon (:) is used to separate a command keyword from a lower-level keyword as shown below:

```
SOURce1:CURRent:PROTection:STATe
```

A semicolon (;) is used to separate two commands within the same subsystem, and can also minimize typing. For example, sending the following command string,

```
SOURce1:VOLTage 20;CURRent 300mA
```

is the same as sending the following two commands:

```
SOURce1:VOLTage 20  
SOURce1:CURRent 1.5
```

Use a colon and a semicolon to link commands from different subsystems. For example, in the following command string, an error is generated if you do not use the colon and semicolon:

```
SYSTem:BEEP;:SOURce1:CURRent 2.5
```

2.4. Querying parameter settings

You can query the value of most parameters by adding a question mark (?) to the command. For example, the following command sets the output voltage to 45.5V:

```
VOLTage 45.5
```

You can query the value by executing:

```
VOLTage?
```

If error is occurred use [SYSTem:ERRor\[:NEXT\]?](#) to get more information about error.

2.5. Using the MIN, MAX, and DEF Parameters

For many commands, you can substitute "MIN" or "MAX" in place of a parameter. In some cases you may also substitute "DEF". For example, consider the following command:

```
[SOURce[<n>]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] {<voltage>|MIN|DEF|MAX|UP|DOWN}
```

Instead of selecting a specific value for the <voltage> parameter, you can substitute MIN to set the voltage to its minimum value, MAX to set the voltage to its maximum value, or DEF to set the voltage to its default value. For list of parameter values see [Section 8.1](#)

2.6. Command and message types

SCPI commands can be divided to **common** and **subsystem** commands.

- Common commands are defined by the IEEE 488.2 standard to perform common interface functions. They begin with an * and consist of three letters (command) or three letters and a ? (query). Description of supported common commands can be found in [Section 4](#)
- Subsystem commands are specific to instrument (PSU in this case) functions. They can be a single command or a group of commands. The groups are comprised of commands that extend one or more levels below the root. See [Section 5](#) for commands that is created in accordance to the SCPI 1999.0 standard. Commands that is not defined by SCPI 1999.0 is labeled "unclassified" and are presented in [Section 6](#)

There are two types of SCPI messages, **program** and **response**.

- A program message consists of one or more properly formatted SCPI commands sent from the controller to the instrument. The message, which may be sent at any time, requests the instrument to perform some action.
- A response message consists of data in a specific SCPI format sent from the instrument to the controller. The instrument sends the message only in response to a **query** header.

2.7. Required Commands

The following commands are required in all SCPI instruments (see SCPI 1999.0 Section 4.2.1):

| Mnemonic | SCPI 1999.0 Command Reference Section | SCPI 1999.0 Syntax and Style Section |
|---------------|---------------------------------------|--------------------------------------|
| :SYSTem | | |
| :ERRor | 21.8 | |
| [:NEXT]? | 21.8.3e | 1996 |
| :VERSion? | 19.16 | 1991 |
| :STATus | 18 | 5 |
| :OPERation | | |
| [:EVENT]? | | |
| :CONDition? | | |
| :ENABle | | |
| :ENABle? | | |
| :QUESTionable | | |
| [:EVENT]? | | |
| :CONDition? | | |
| :ENABle | | |
| :ENABle? | | |
| :PRESet | | |

2.7.1. Base functionality for the Power supply instrument class

| SCPI Command | Description |
|-----------------|---|
| OUTPut | |
| [:STATe] <bool> | Enables the specified output channel(s) |
| [SOURce[<n>]] | |
| CURRent | |

```

[:LEVel]
    [:IMMediate][:AMPLitude] <current>    Sets the output current
VOLTage
    [:LEVel]
    [:IMMediate][:AMPLitude] <voltage>    Sets the output voltage

```

All SCPI power supplies shall implement the status reporting structure. STATUS Subsystem defines the commands which shall be used to control the status reporting structure. For a power supply, the bits of interest in the QUESTIONable status structure are VOLTage and CURRENT. When a power supply is operating as a voltage source, bit 1 (CURRENT) shall be set. When a power supply is operating as a current source, bit 0 (VOLTage) shall be set. When the output is unregulated, both bits shall be set (for example, while the output is changing to a new programmed value).

2.8. Multiple Commands in a Message

Multiple SCPI commands can be combined and sent as a single message with one message terminator. There are two important considerations when sending several commands within a single message:

- Use a semicolon to separate commands within a message.
- There is an implied header path that affects how commands are interpreted by the PSU.

The header path can be thought of as a string that gets inserted before each command within a message. For the first command in a message, the header path is a null string. For each subsequent command the header path is defined as the characters that make up the headers of the previous command in the message up to and including the last colon separator. An example of a message with two commands is:

```
OUTPut:STATe ON,CH1;PROTection:CLEAr CH1
```

which shows the use of the semicolon separating the two commands, and also illustrates the header path concept. Note that with the second command, the leading header OUTPut was omitted because after the OUTPut:STATe ON command, the header path became defined as OUTPut and thus the instrument interpreted the second command as:

```
OUTPut:PROTection:CLEAr CH1
```

In fact, it would have been syntactically incorrect to include the OUTPut explicitly in the second command, since the result after combining it with the header path would be:

```
OUTPut:OUTPut:PROTection:CLEAr CH1
```

which is incorrect.

You can combine common commands (IEEE488) with subsystem commands in the same message. Treat the common command as a message unit by separating it with a semicolon (the message unit separator). Common commands do not affect the header path; you may insert them anywhere in the message.

```
*TST?;SYSTem:ERRor?
```

2.9. Moving Among Subsystems

In order to combine commands from different subsystems, you need to be able to reset the header path to a null string within a message. You do this by beginning the command with a colon (:), which discards any previous header path. For example, you could clear the output protection and check the status of the Operation Condition register in one message by using a root specifier as follows:

```
OUTPut:PROTection:CLEAr CH1;:STATus:OPERation:CONDition? CH1
```

The following message shows how to combine commands from different subsystems as well as within the same subsystem:

```
VOLTage:LEVel 7.5,CH1;PROTection 10,CH1;:CURRENT:LEVel 0.5,CH1
```

Note the use of the optional header LEVel to maintain the correct path within the subsystems, and the

use of the root specifier to move between subsystems.

2.10. SCPI parameter types

The SCPI language defines several different data formats to be used in program messages and response messages:

Numeric Commands that require numeric parameters will accept all commonly used representations of numbers like integer (also known as NR1 format specified in ANSI X3.42-1990) or decimal representations of numbers including optional signs, decimal points (NR2 format), and scientific notation (i.e. 10E3 or NR3 format). Special values for numeric parameters like MINimum, MAXimum, and DEFault are also accepted. You can also send engineering unit suffixes (V, A, or SEC) with numeric parameters. If only specific numeric values are accepted, the PSU will automatically round the input numeric parameters. The following command uses a numeric parameter:

```
VOLT:STEP {<step>}
```

Discrete Used to program settings that have a limited number of values such as BUS and IMM or CH1 and CH2. Query responses will always return the short form in all uppercase letters. The following command uses discrete parameters:

```
CAL:CURR:LEV {MIN|MID|MAX}
```

Boolean Represent a single binary condition that is either true or false. For a false condition, the PSU will accept OFF or 0. For a true condition, the PSU will accept ON or any nonzero value (i.e. 1 but also 2.34 or -3). When you query a Boolean setting, the PSU will always return 0 or 1. The following command uses a Boolean parameter:

```
OUTP {OFF|ON}
```

String Can contain virtually any set of ASCII characters. A string must begin and end with matching quotes, either with a single quote or with a double quote. You can include the quote delimiter as part of the string by typing it twice without any characters in between. The following command uses a string parameter:

```
CAL:REM <quoted string>
```

3. Registers and queues

SCPI requires the status mechanism described in Section 11 of IEEE 488.2, including full implementation of the status register structure. Summary of implemented registers structure for the PSU is shown on Fig. 1. (commands used to access registers are written in parentheses).

All SCPI instruments have to implement status registers in the same way. The status system records various instrument conditions in the following register groups:

- the Status Byte register,
- the Standard Event register,
- the QUEStionable Status register group, and
- the OPERation Status register group.

The Status Byte register records high-level summary information reported in the other register groups. Message interchanging between Controller and Instrument is accomplished by using input buffer and Output queue and Error queue. The length of the Input buffer is 48 characters. Both Output and Error queue can handle up to 20 messages.

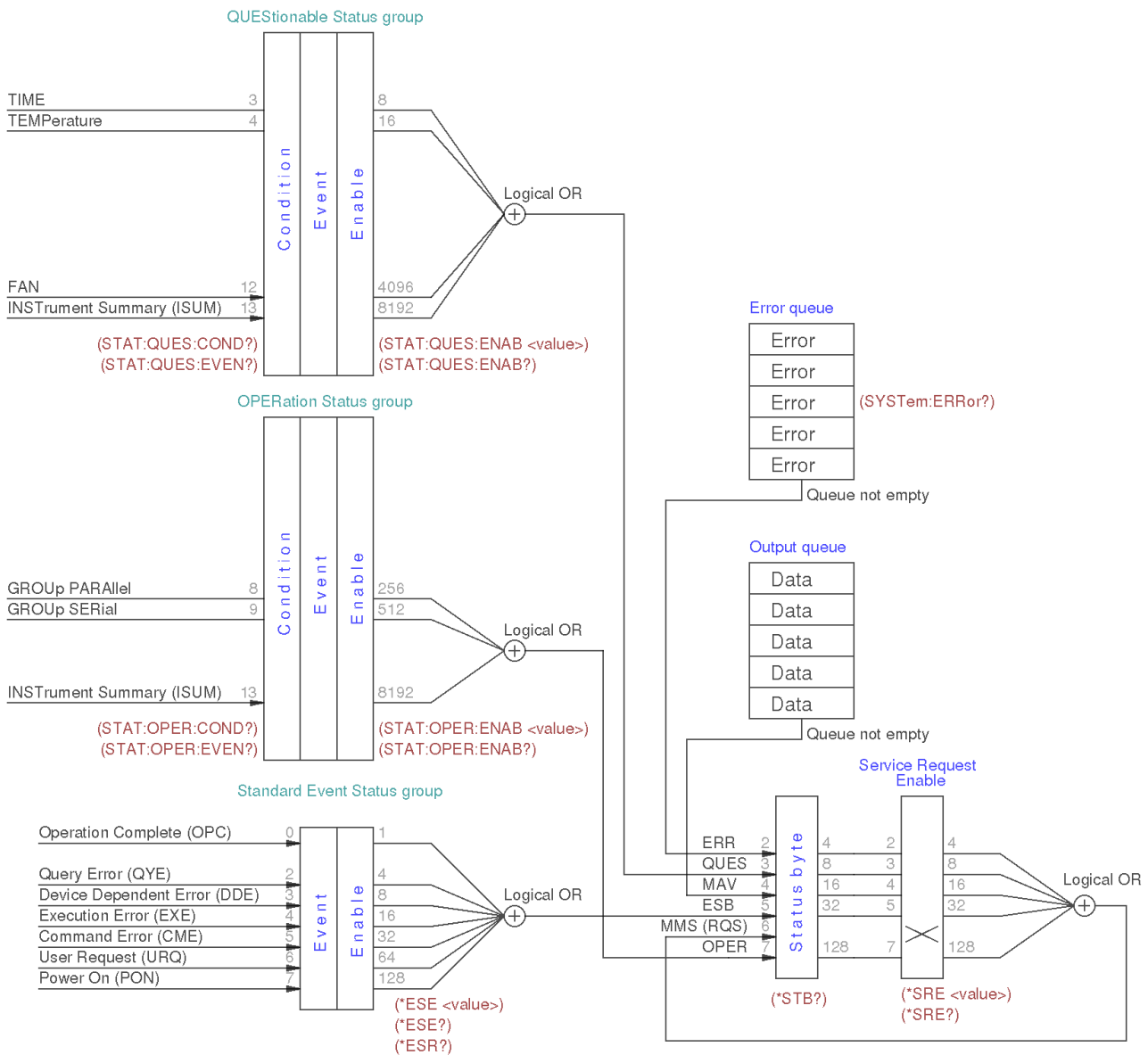


Fig. 1: Summary of status structure registers

3.1. Standard Event Status Register

An status register group is consist of Condition, Event and Enable registers (see Fig. 1):

- The Condition register is a read-only register, which holds the live (unlatched) operational status of the instrument. Reading the Condition register does not clear it.
- The Event register is a read-only that reports defined conditions within the PSU. Bits in an event register are latched. Once an event bit is set, subsequent state changes are ignored. Bits in the Event register are automatically cleared by a query of that register (such as *ESR? or STATus:QUESTionable:EVENT?) or by sending the *CLS (clear status) command. A reset (*RST) or device clear will *not* clear bits in event registers (See [Section 8.2](#)). Querying an event register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.
- The ENABLE register is used to define which bits of the Event Status register will latch ESB (bit 5) of the Status byte register.

An error status (bit 2, 3, 4 or 5) records one or more errors in the PSU error queue. The SYSTem:ER-Ror? command can be used to read the error queue.

Implementation of the Standard Event Status register follows IEEE 488.2 Section 11.5.1.1:

| Bit | Decimal value | Description |
|-----|---------------|--|
| 0 | 1 | Operation Complete (OPC) – This event bit is generated in response to the *OPC command. It indicates that the PSU has completed all selected pending operations (including *OPC). |
| 1 | 2 | Not used |
| 2 | 4 | Query ERROR (QYE) – Query Errors are detected by the Output Queue Control. This event bit indicates that either <ul style="list-style-type: none"> • An attempt is being made to read data from the Output Queue when no output is either present or pending, or • Data in the Output Queue has been lost. Events that generate Query Errors do not generate Execution Errors, Command Errors, or Device-Specific Errors. |
| 3 | 8 | Device-Specific ERROR (DDE) – This event bit indicates that an error has occurred that is neither a Command Error, a Query Error, nor an Execution Error. A Device-Specific Error is any executed device operation that did not properly complete due to some condition, such as over-range, a self-test or calibration error. Following a Device-Specific Error, the PSU will continue to process the input stream. Events that generate Device-Specific Errors do not generate Command Errors, Query Errors, or Execution Errors. |
| 4 | 16 | Execution ERROR (ERR) – This event bit indicates that: <ul style="list-style-type: none"> • A <PROGRAM DATA> element following a header was evaluated by the PSU as outside of its legal input range or is otherwise inconsistent with the PSU's capabilities. • A valid program message could not be properly executed due to some PSU condition. Following an Execution Error, the PSU will continue parsing the input stream. Execution Errors will be reported by the PSU after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, will not be reported as an Execution Error. Events that generate Execution Errors do not generate Command Errors, Query Errors, or Device-Specific Errors. |
| 5 | 32 | Command ERROR (CME) – Command Errors are detected by the parser. This |

event bit indicates that one of the following events has occurred:

- An IEEE 488.2 syntax error has been detected by the parser. That is, a controller-to-device message was received that is in violation of this standard. Possible violations include a data element that violates the device listening formats or whose type is unacceptable to the device (see also IEEE 488.2 Section 7.1.2.2).
- A semantic error has occurred indicating that an unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented common commands described in [Section 4](#)
- A Group Execute Trigger (GET) was entered into the Input Buffer inside of a <PROGRAM MESSAGE> (see also IEEE 488.2 Section 6.1.6.1.1 and 6.4.3.).

When the PSU detects a Command Error, parser synchronization may be lost.

When a Command Error is detected, any prior parsable elements of the same <PROGRAM MESSAGE> will be executed. That is also true for all parsable elements that follows after detected Command Error.

The Command Error bit not be set to report any other device-specific condition. Events that are reported as Command Errors cannot be reported as Execution Errors, Query Errors, or Device-Specific Errors.

| | | |
|------|-----|---|
| 6 | 64 | User Request (URQ) – This event bit indicates that the PSU input device (TFT Touch screen) has been for any reason activated. The setting of this event-bit occur regardless of the IEEE 488.1 Remote/Local state of the device (not supported in M1) |
| 7 | 128 | Power On (PON) – This event bit indicates that an off-to-on transition has occurred in the device's power supply. See also SYSTem:POWer. |
| 8-15 | – | Not used, always zero |

3.2. Status Byte Register

The Status Byte summary register reports conditions from the other status registers (see Fig. 1). Query data that is waiting in the PSU's output buffer is immediately reported through the "Message Available" (MAV) bit (bit 4) of the Status Byte register. Bits in the summary register are NOT latched. Clearing an event register will clear the corresponding bits in the Status Byte summary register. Reading all messages in the output buffer, including any pending queries, will clear the message available bit (MAV). The Status Byte summary register is cleared when the *CLS (clear status) command has been executed.

The Status Byte enable register (request service) is cleared when the *SRE 0 command has been executed.

Querying the Standard Event register (*ESR? command) will clear only bit 5 (ESR) in the Status Byte summary register. For example, 24 (8 + 16) is returned when you have queried the status of the Status Byte register, QUES and MAV conditions have occurred.

| Bit | Decimal value | Description |
|-----|---------------|---|
| 0-1 | – | Not used, always zero |
| 2 | 4 | ERR – Error queue bit indicates that one or more errors have been stored in the Error queue. |
| 3 | 8 | QUES – One or more bits are set in the QUESTionable Status register (bits must be enabled in the enable register). |
| 4 | 16 | MAV – The Message Available bit indicates whether or not the Output Queue is empty. Whenever the device is ready to accept a request by the controller to output data bytes, the MAV is TRUE. The MAV is FALSE when the Output Queue is empty. This bit is used to synchronize information exchange with the controller. The con- |

troller can, for example, send a query command to the device and then wait for MAV to become TRUE.

- | | | |
|---|-----|--|
| 5 | 32 | ESB – One or more bits are set in the Standard Event register (bits must be enabled in the enable register, see *ESE command). |
| 6 | 64 | MMS – Master Status summary bit indicates that one or more bits are set in the Status Byte Register (bits must be enabled, see *SRE command). Also used to indicate a request for service (RQS). |
| 7 | 128 | OPER – One or more bits are set in the OPERation Status register. |

3.3. OPERation Status Register

The OPERation status register contains conditions which are part of the instrument's normal operation. Each channel of the PSU is considered as separate "instrument". The two logical outputs (channels) of the PSU include an INSTRument summary status register and an individual instrument ISUMmary register for each logical output.

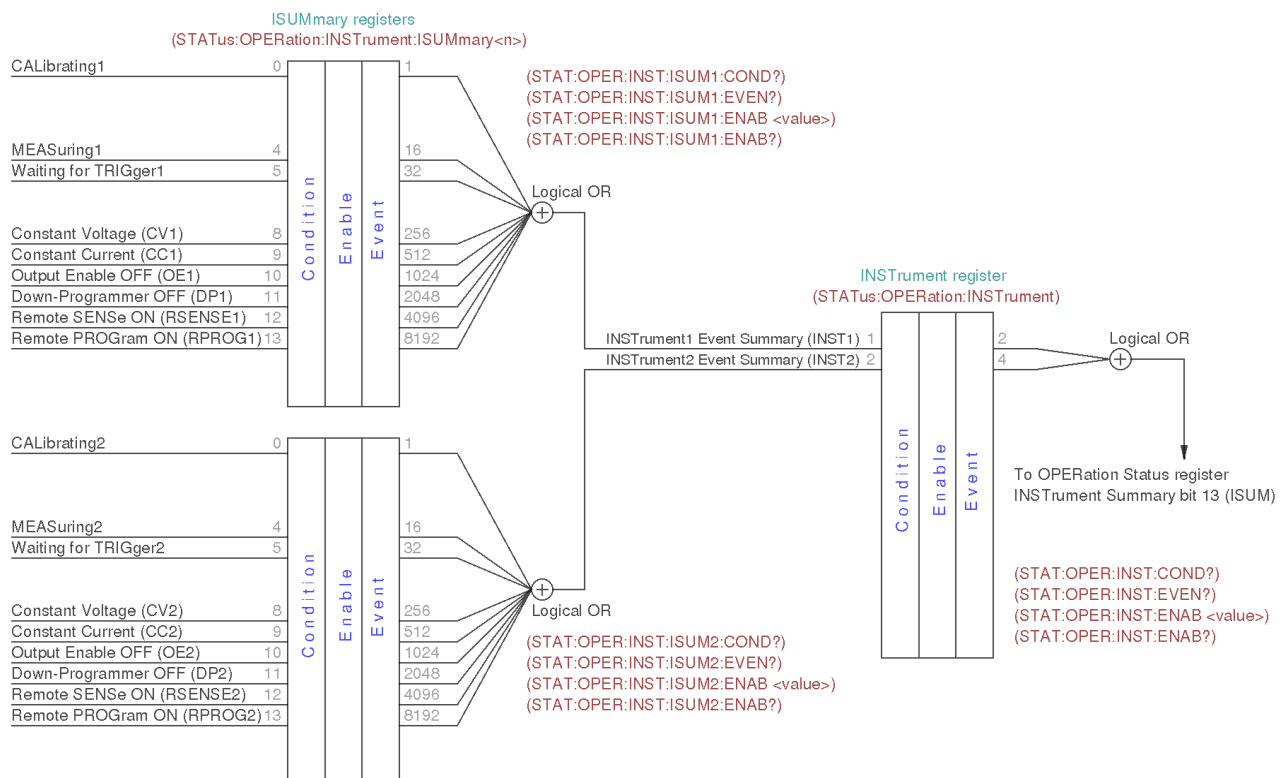


Fig. 2: OPERation Status registers summary

The bit definition of OPERation Status register shown on Fig.1.:

| Bit | Decimal value | Description |
|-------|---------------|---|
| 0-7 | – | Not used, always zero |
| 8 | 256 | GROUp PARAllel indicate that PSU's channels are connected in parallel. |
| 9 | 512 | GROUp SERIal indicate that PSU's channels are connected in serial. |
| 10-12 | – | Not used, always zero |
| 13 | 8192 | INSTRument Summary Bit – One of n multiple logical instruments is reporting OPERational status. |
| 14-15 | – | Not used, always zero |

The Event Status Enable register is cleared when the STAT:EVEN:ENAB 0 command is executed. The

*CLS command can be also used to clear the register.

3.3.1. Operation INSTRument Status register

The bit definition of OPERation INSTRument Status register shown on Fig.2.:

| Bit | Decimal value | Description |
|------|---------------|---|
| 0 | – | Not used, always zero |
| 1 | 2 | INST1 – Instrument1 summary bit indicate that one or more bits are changed in the Channel 1 OPERation INSTRument Summary register |
| 2 | 4 | INST2 – Instrument2 summary bit indicate that one or more bits are changed in the Channel 2 OPERation INSTRument Summary register |
| 3-15 | – | Not used, always zero |

3.3.2. Operation Instrument SUMmary status register

The ISUMmary registers report to the INSTRument register, which in turn reports to bit 13 of the Operation Status register. This is illustrated on Fig. 2. Using such a status register configuration allows a status event to be cross- referenced by output channel and type of event. The INSTRument register indicates which channel(s) have generated an event. The ISUMmary register represent a pseudo-operation Status register for a particular logical output.

The bit definition of OPERation INSTRument ISUMmary Status register shown on Fig.2.:

| Bit | Decimal value | Description |
|-------|---------------|--|
| 0 | 1 | CALibrating – Channel is performing calibration |
| 1-3 | – | Not used, always zero |
| 4 | 16 | MEASuring – Channel is performing measurement (not supported in M1) |
| 5 | 32 | Waiting for TRIGger – Channel is waiting for the trigger event (not supported in M1) |
| 6-7 | – | Not used, always zero |
| 8 | 256 | CV – Channel is entered CV operation mode |
| 9 | 512 | CC – Channel is entered CC operation mode |
| 10 | 1024 | OE – Output is switched off |
| 11 | 2048 | DP – Down-programmer is switched off |
| 12 | 4096 | RSENSE – Remote voltage sense is switched on |
| 13 | 8192 | RPROG – Remote voltage programming is switched on |
| 14-15 | – | Not used, always zero |

3.4. QUEStionable Status Register

The Questionable Status register provides information about unexpected operations of the PSU. Each channel of the PSU is considered as separate "instrument". The two logical outputs (channels) of the PSU include an INSTRument summary status register and an individual instrument ISUMmary register for each logical output.

The ISUMmary registers report to the INSTRument register, which in turn reports to bit 13 of the Questionable Status register. This is illustrated on Fig. 3. Using such a status register configuration allows a status event to be cross-referenced by output channel and type of event. The INSTRument register indicates which channel(s) have generated an event. The ISUMmary register represent a pseudo-Questionable Status register for a particular logical output.

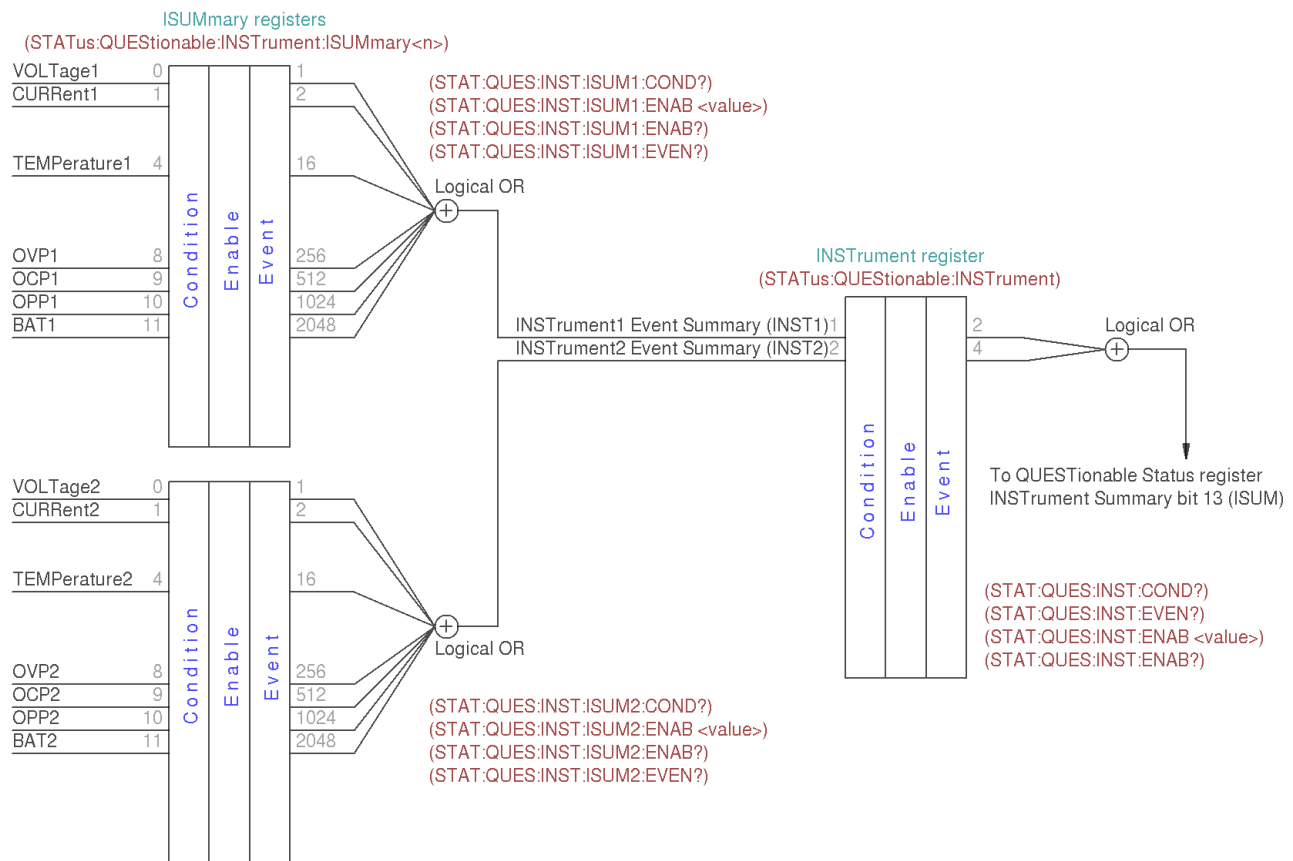


Fig. 3: QUESTionable INSTRument registers summary

For example, if one of the two channels is in constant voltage (CV) mode and due to an overload loses regulation, bit 13 is set (latched). To read the register, the command `STATus:QUESTionable?` is required. To make use of bit 13 (ISUM), enable register must be correctly set. The command `STAT:QUES:INST:ENAB 6 (2 + 4)` has to be sent to enable the Questionable instrument register, followed by the command `STAT:QUES:INST:ISUM<n>:ENAB 19` for each channel to enable the QUESTionable INSTRument SUMmary register, where *n* is 1 or 2.

Bit definition for QUESTionable Status register shown on Fig.1.:

| Bit | Decimal value | Description |
|-------|---------------|--|
| 0-2 | – | Not used, always zero |
| 3 | 8 | TIME – indicate abnormal time/date situation due to RTC failure or conflict between current and time/date retrieved from the stored configuration. |
| 4 | 16 | TEMPerature – temperature measurement that use the temperature sensor on the Arduino Shield board require attention (i.e. over-temperature condition is detected, sensor is not functional, etc.). <i>Do not confuse this sensor with that are connected to a PSU channels.</i> |
| 5-11 | – | Not used, always zero |
| 12 | 4096 | FAN – cooling fan failure detected |
| 13 | 8192 | INSTRument summary, is described later in this chapter in association with multiple logical instruments. |
| 14-15 | – | Not used, always zero |

The Questionable Status Enable register is cleared when the `STAT:QUES:ENAB 0` command is executed. The `*CLS` command can be also used to clear the register.

3.4.1. Questionable INSTRument Status register

Bit definition for QUESTionable INSTRument register:

| Bit | Decimal value | Description |
|------|---------------|--|
| 0 | – | Not used, always zero |
| 1 | 2 | INST1 – Instrument1 summary bit indicate that one or more bits are changed in the Channel 1 OPERation INSTRument Summary register. |
| 2 | 4 | INST2 – Instrument2 summary bit indicate that one or more bits are changed in the Channel 2 OPERation INSTRument Summary register. |
| 3-15 | – | Not used, always zero |

3.4.2. Questionable Instrument SUMmary status register

There are two questionable instrument summary registers, one for each PSU output. These registers provide information about voltage and current regulation.

Bit definition for QUESTionable INSTRument SUMmary register:

| Bit | Decimal value | Description |
|-------|---------------|--|
| 0 | 1 | VOLTage – This bit is set when the voltage becomes unregulated, therefore a channel enters CC operation mode. <i>If the over-voltage protection (OVP) is activated (see VOLTage:PROTection:STATe) channel output will be switched off.</i> |
| 1 | 2 | CURRent – This bit is set when the current becomes unregulated, therefore a channel enters CV operation mode. <i>If the over-current protection (OCP) is activated (see CURRent:PROTection:STATe) channel output will be switched off.</i> |
| 2-3 | – | Not used, always zero |
| 4 | 16 | TEMPerature – one of the temperature sensor (heatsink or battery) on the channel power board require attention (i.e. over-temperature condition is detected, sensor is not functional, etc.). <i>Do not confuse this sensor with one that is connected to the Arduino shield board.</i> |
| 5-7 | – | Not used, always zero |
| 8 | 256 | OVP – Over-voltage protection is activated. The query VOLT:PROT:TRIP? returns value of this bit. See also STAT:QUES. |
| 9 | 512 | OCP – Over-current protection is activated. The query CURR:PROT:TRIP? returns value of this bit. See also STAT:QUES. |
| 10 | 1024 | OPP – Over-power protection is activated. The query POW:PROT:TRIP? returns value of this bit. See also STAT:QUES. |
| 11 | 2048 | FAN – cooling fan failure is detected. See also STAT:QUES. |
| 12 | 4096 | BAT – Over-temperature protection triggered by battery temperature sensor is activated. The query TEMP:PROT:TRIP? BAT<n> returns value of this bit. See also STAT:QUES. |
| 13-15 | – | Not used, always zero |

Please note here that CURRent bit is use for questionable Voltage operating mode and vice versa.

If 0 and 1 bits is true that indicate neither the voltage nor the current is regulated (so-called unregulated or UR mode), and both bits false indicate the PSU channel are off.

To read the register for each PSU channel, the command STAT:QUES:INST:ISUM[<n>]? has to be send, where [<n>] is 1 or 2. If [<n>] is not specified the currently selected channel is used.

Use STAT:QUES:INST:ISUM<n>:COND? to determine operating mode (CV or CC) for the PSU channel

(where n is 1 or 2 depending on the output).

The Questionable Status event register is cleared with:

- the *CLS (clear status) command or
- the event register is queried using the STAT:QUES? (status questionable event register) command.

3.5. Error queue

The error queue contains items that include a numerical and textual description of the error or event.

The <Error/event_number> is a unique integer in the range [-32 768, 32 767]. All positive numbers are instrument-dependent. All negative numbers are reserved by the SCPI standard with certain standard error/event codes. The value, zero, is also reserved to indicate that no error or event has occurred.

The second parameter of the full response is a quoted string containing an <Error/event_description>. Each <Error/event_number> has a unique and fixed <Error/event_description> associated with it. An example:

```
-113, "Undefined header"
```

The maximum string length of <Error/event_description> plus <Device-dependent_info> is 255 characters. List of all error/event messages can be found in [Section 7](#) of this document.

As errors and events are detected, they are placed in a queue. This queue is first in, first out. If the queue overflows, the last error/event in the queue is replaced with error:

```
-350, "Queue overflow"
```

Any time the queue overflows, the least recent errors/events remain in the queue, and the most recent error/event is discarded. Reading an error/event from the head of the queue removes that error/event from the queue, and opens a position at the tail of the queue for a new error/event, if one is subsequently detected.

If the error queue is not empty, bit 2 of the Instrument Summary Status Register is set. A query returns only the oldest error code and associated error description information from the error queue. To return all error codes and associated description information, use repetitive queries until an error value of zero is returned, or until bit 2 of the status register is 0.

The error queue is cleared when any of the following occur (IEEE 488.2, section 11.4.3.4):

- Upon power up
- Upon receipt of a *CLS command
- Upon reading the last error message from the queue

4. Common command reference

This section summarizes the mandatory subset of IEEE 488.2 commands that is a requirement for any SCPI compliant instrument.

| Common command | Description |
|--|--|
| *CLS | Clears all status data structures |
| *ESE {<value>} | Programs the Standard Event Status Enable register bits |
| *ESR? | Reads the Standard Event Status Register |
| *IDN? | Returns the UNIQUE identification of the PSU |
| *OPC | Operation Complete Command used for program synchronization |
| *RCL {<profile>} | Recalls the PSU state stored in the specified storage location |
| *RST | Reset PSU to the initial state |
| *SAV {<profile>} | Stores the current PSU state in the specified storage location |
| *STB? | Reads the Status Byte register |
| *TST? | Returns Self-Test results |

4.1. *CLS

| | |
|------------------|--|
| Syntax | *CLS |
| Description | <p>Clear Status Command. This command clears all status data structures in the PSU:</p> <ul style="list-style-type: none">• Standard Event Status Register• OPERation Event Status Register• QUEStionable Event Status Register• Error/Event Queue <p>The corresponding condition and enable registers are unaffected. If *CLS immediately follows a program message terminator (<NL>), then the output queue and the MAV bit are also cleared.</p> |
| Return | None |
| Related Commands | *ESR? STATus:OPERation[:EVENT] STATus:OPERation:INSTRument[:EVENT] STATus:OPERation:INSTRument:ISUMmary[<n>][:EVENT] STATus:QUEStionable[:EVENT] STATus:QUEStionable:INSTRument[:EVENT] STATus:QUEStionable:INSTRument:ISUMmary[<n>][:EVENT] SYSTem:ERRor |

4.2. *ESE

| | | | | |
|-------------|---|------|-------|---------|
| Syntax | *ESE {<value>} *ESE? | | | |
| Description | <p>Standard Event Status Enable Command. This command programs the Standard Event Status Enable register bits. The programming determines which events of the Standard Event Status Event register (see *ESR?) are allowed to set the ESB (Event Summary Bit) of the Status Byte register. A 1 in the bit position enables the corresponding event. All of the enabled events of the Standard Event Status Event Register are logically ORed to cause the Event Summary Bit (ESB) of the Status Byte Register to be set.</p> <p>A STATus:PRESet command does not clear the bits in the Status Byte register. See also Section 3.1 in this document.</p> | | | |
| Parameters | Name | Type | Range | Default |

| | | | | |
|------------------|--|---------|---|------|
| | <value> | Numeric | 0 – 255 (A decimal value which corresponds to the binary-weighted sum of the bits in the register. See also table in Section 3.1). | None |
| Return | The query reads the enable register and returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register. | | | |
| Usage example | <p>To enable bit 2 (decimal value = 4), bit 3 (decimal value = 8), and bit 7 (decimal value = 128), the corresponding decimal value would be 140 (4 + 8 + 128):</p> <pre>*ESE 140</pre> <p>Read value of the Standard Event Status Enable register:</p> <pre>*ESE? 140</pre> | | | |
| Errors | -200, "Execution error" | | | |
| Related Commands | <pre>*CLS *RST STATus:PRESet</pre> | | | |

4.3. *ESR?

| | |
|---------------|--|
| Syntax | *ESR? |
| Description | Standard Event Status Register (see Section 3.2) Query. Reading the Standard Event Status Event register clears it. |
| Return | The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register. |
| Usage example | <p>If ERRor (bit 2) is set:</p> <pre>ESR? 4</pre> |

4.4. *IDN?

| | |
|---------------|---|
| Syntax | *IDN? |
| Description | Identification query for the UNIQUE identification of the PSU. (see also IEEE 488.2 10.14). |
| Return | The following system parameters will be displayed: <vendor>, <model>, <serial number>, <firmware>. The <model> include information about used CPU in brackets and could be Mega, Due or Simulator. More information about simulator could be found in Section 9 . |
| Usage example | <pre>*IDN? EEZ, PSU 2/50/03 (Due), 00001, M1.0.96</pre> |

4.5. *OPC

| | |
|-------------|--|
| Syntax | *OPC *OPC? |
| Description | <p>Operation Complete Command. The command is mainly used for program synchronization. It causes the PSU to set the OPC bit (bit 0) of the Standard Event Status register when the PSU has completed all pending operations sent before *OPC. Pending operations are complete when:</p> <ul style="list-style-type: none"> All commands sent before *OPC, including paralleled commands, have been completed. Most commands are sequential and are completed before the next command is executed. Commands that affect output voltage, current, or state, |

relays, and trigger actions are executed in parallel with subsequent commands. *OPC provides notification that all parallel commands have completed.

- All triggered actions are completed.

Query whether the current operation is completed and the query returns 1.
See also IEEE 488.2 Section 12.5 – 12.8.

| | |
|---------------|--|
| Return | Query causes the PSU to place a 1 in the output buffer when all pending operations are completed. *OPC? does not suspend processing of commands. |
| Usage example | <p>*OPC?</p> <p>1</p> <p>if current operation is not completed:</p> <p>*OPC?</p> <p>0</p> |

4.6. *RCL

| | | | | |
|------------------|--|---------|-------|---------|
| Syntax | *RCL {<profile>} | | | |
| Description | <p>This command recalls the PSU state stored in the specified storage location. The PSU has ten storage locations in non-volatile memory to store PSU states. It is not possible to recall the PSU state from a storage location that is empty or was deleted. When the firmware is started for the first time, storage locations 1 through 9 are empty (location 0 has the power-on state).</p> <p><i>The PSU uses location 0 to automatically hold the state of the PSU at power down.</i></p> | | | |
| Parameters | Name | Type | Range | Default |
| | <profile> | Numeric | 0 – 9 | None |
| Return | None | | | |
| Usage example | *RCL 2 | | | |
| Errors | | | | |
| Related Commands | <p>*SAV</p> <p>MEMory:STATe:DELeTe</p> <p>MEMory:STATe:RECall:AUTO</p> <p>MEMory:STATe:RECall:SELEct</p> <p>SYSTem:POWeR</p> | | | |

4.7. *RST

| | | | | |
|------------------|---|--|--|--|
| Syntax | *RST | | | |
| Description | <p>Reset Command. Restore the PSU to initial state (as predefined in the PSU firmware, see Section 8.2) and clear the error queue. The reset command explicitly will NOT affect calibration data nor any of saved configuration profiles (0 to 9). When *RST is issued all output are set to OFF, and voltage and current are programmed to 0. Power up sequence is started. All SPI peripherals are reinitialize except Ethernet controller if an active Ethernet connection exists.</p> | | | |
| Return | None | | | |
| Usage example | <p>*RST</p> <p>MEMory:RECall:AUTO</p> <p>SYSTem:POWeR</p> | | | |
| Related Commands | <p>*RST</p> <p>*SAV</p> | | | |

MEMory:STATe:CATalog?

4.8. *SAV

Syntax ***SAV {<profile>}**

Description This command stores the current instrument state in the specified storage location. Any state previously stored in the same location is overwritten without generating any errors. The PSU has nine storage locations in non-volatile memory available to user to store current PSU states. The following channel and system parameters will be stored in the non-volatile memory:

- Calibration status ([CALibration:STATe](#))
- Output enable state ([OUTPut\[:STATe\]](#))
- Remote sense state ([\[SOURce\[<n>\]\]:VOLTage:SENSe\[:SOURce\]](#))
- Output voltage ([\[SOURce\[<n>\]\]:VOLTage](#))
- Output voltage step ([\[SOURce\[<n>\]\]:VOLTage:STEP](#))
- OVP status ([\[SOURce\[<n>\]\]:VOLTage:PROTection:STATe](#))
- OVP delay ([\[SOURce\[<n>\]\]:VOLTage:PROTection:DELAy](#))
- Output current ([\[SOURce\[<n>\]\]:CURRent](#))
- Output current step ([\[SOURce\[<n>\]\]:CURRent:STEP](#))
- OCP status ([\[SOURce\[<n>\]\]:CURRent:PROTection:STATe](#))
- OCP delay ([\[SOURce\[<n>\]\]:CURRent:PROTection:DELAy](#))
- OPP level ([\[SOURce\[<n>\]\]:POWER:PROTection\[:LEVel\]](#))
- OPP status ([\[SOURce\[<n>\]\]:POWER:PROTection:STATe](#))
- OPP delay ([\[SOURce\[<n>\]\]:POWER:PROTection:DELAy](#))
- OTP level ([SYSTem:TEMPerature:PROTection\[:HIGH\]\[:LEVel\]](#))
- OTP status ([SYSTem:TEMPerature:PROTection\[:HIGH\]:STATe](#))
- OTP delay ([SYSTem:TEMPerature:PROTection\[:HIGH\]:DELAy](#))
- Power on state ([SYSTem:POWER](#))
- Simulator load value ([SIMUlator:LOAD](#))
- Simulator load connection ([SIMUlator:LOAD:STATe](#))

You can assign a user-defined name to each of locations 1 through 9 using the [MEMory:STATe:NAME](#) command.

A reset ([*RST](#) command) does not affect the configurations stored in memory. Once a state is stored, it remains until it is overwritten using this command or specifically deleted using the [MEMory:STATe:DELeTe](#) command.

The PSU uses location 0 to automatically hold the state of the PSU at power down.

| Parameters | Name | Type | Range | Default |
|------------------|---|---------|-------|---------|
| | <profile> | Numeric | 1 – 9 | None |
| Return | None | | | |
| Usage example | *SAV 2 | | | |
| Related Commands | *RCL *RST MEMory:STATe:CATalog? MEMory:STATe:NAME MEMory:STATe:DELeTe | | | |

4.9. *STB?

Syntax ***STB?**

Description Read Status Byte Query. This query reads the Status Byte register (see [Section 3.2](#)), which contains the status summary bits and the Output Queue MAV bit. The Status

Byte is a read-only register and the bits are not cleared when it is read.

A serial poll also returns the value of the Status Byte register, except that bit 6 returns Request for Service (RQS) instead of Master Status Summary (MSS). A serial poll clears RQS, but not MSS. When MSS is set, it indicates that the PSU has one or more reasons for requesting service.

| | |
|------------------|---|
| Return | The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register. |
| Usage example | If OPER (bit 7) is set: *STB? 128 |
| Related Commands | *SRE |

4.10. *TST?

| | |
|------------------|--|
| Syntax | *TST? |
| Description | <p>Self-Test Query. The self-test query causes an internal self-test and places a response into the Output Queue indicating whether or not the PSU completed the self-test without any detected errors.</p> <p><i>Ensure that all terminal connections are removed while the internal self-test is being performed.</i></p> <p><i>If an active Ethernet connection exists, testing of Ethernet controller will be skipped. You can use DIAGnostic[:INFormation]:TEST? for query detailed report of the latest self-test.</i></p> |
| Return | 0 or 1 depends of the self-test results. See also DIAGnostic[:INFormation]:TEST? . |
| Usage example | <p>If all tests passed:</p> <p>*TST? 0</p> <p>If one or more tests failed:</p> <p>*TST? 1</p> |
| Related Commands | <p>DIAGnostic[:INFormation]:TEST?</p> <p>SYSTem:BEEP:STATe</p> |

5. Subsystem command reference

This section summarizes the Standard Commands for Programmable Instruments (SCPI) available to program the PSU over the remote interface.

- [CALibrate](#)
- [DIAGnostic](#)
- [INSTrument](#)
- [MEASure](#)
- [MEMory](#)
- [OUTPut](#)
- [SOURce](#)
- [STATus](#)
- [SYSTem](#)

5.1. CALibrate

This subsystem provides commands for the PSU calibration. Only one channel can be calibrated at a time. If calibration mode has not been enabled with CALibrate:STATe, the calibration commands will generate an error. Use CALibrate:SAVE to save any changes, otherwise all changes will be lost on exit from calibration mode. Within the same calibration session both output voltage and current can be calibrated for the currently selected channel.

| SCPI Command | Description |
|---|--|
| CALibrate[:MODE] {<bool>, <password>} | Enables/disables calibration mode |
| :CLEar {<password>} | Clears all calibration parameters |
| :CURRent | |
| [:DATA] {<new value>} | Enters the calibration value |
| :LEVel {<level>} | Calibrates the output current programming |
| :PASSword | |
| :NEW {<old>, <new>} | Changes calibration password |
| :REMark {<string>} | Saves calibration information |
| :SAVE | Saves the new cal constants in non-volatile memory |
| :STATe {<bool>, <password>} | Enables calibration parameters |
| :VOLTage | |
| [:DATA] {<new value>} | Enters the calibration value |
| :LEVel {<level>} | Calibrates the output voltage programming |

5.1.1. CALibration[:MODE]

Syntax [CALibration\[:MODE\] {<bool> <password>}](#)
 [CALibration\[:MODE\]?](#)

Description This command enables or disables calibration mode. Calibration mode must be enabled for the channel to accept any calibration commands. The first parameter specifies the ON (1) or OFF (0) state. The second parameter is the password. Successful execution of this command set both output VOLTage and CURRent of the selected channel to the MINimum value (see [Section 8.1](#)). Execution of this command also affects bit 0 (CALibrating) of the Operation Instrument Isummary register (see [Section 3.3.2](#)).

If both voltage and current calibration parameters exists on calibration mode exit (CALibration[:MODE] OFF) the CALibration:STATe ON command will automatically follows.

| | | | | |
|------------|------|------|-------|---------|
| Parameters | Name | Type | Range | Default |
|------------|------|------|-------|---------|

| | | | | |
|------------------|--|---------------|--------------------|----------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| | <password> | Quoted string | 4 to 16 characters | "eezpsu" |
| Return | The returned parameter is 0 (OFF) or 1 (ON). | | | |
| Usage example | See Section 10.2 | | | |
| Errors | 102, "Invalid cal password" 104, "Bad sequence of calibration commands" | | | |
| Related Commands | CALibration:STATe DIAGnostic[:INfOrmation]:OTIME? | | | |

5.1.2. CALibration:CLEar

| | | | | |
|------------------|---|---------------|--------------------|----------|
| Syntax | CALibration:CLEar {<password>} | | | |
| Description | Clear all calibration parameters stored in the non-volatile memory for the currently selected channel. After successful execution of this command CALibration:STATe will be set to OFF (0) and further usage of the calibration data will be disabled. This command will be also filled calibration remark with the date and note that calibration data has been cleared. | | | |
| Parameters | Name | Type | Range | Default |
| | <password> | Quoted string | 4 to 16 characters | "eezpsu" |
| Return | None | | | |
| Usage example | CAL:STAT? 1 CAL:CLE CAL:STAT? 0 DIAG:CAL? "remark=2015-11-02 Not calibrated", "u_cal_params_exists=0", "i_cal_params_exists=0" | | | |
| Errors | 102, "Invalid cal password" | | | |
| Related Commands | CALibration:STATe DIAGnostic[:INfOrmation]:CALibration? | | | |

5.1.3. CALibration:CURREnt[:DATA]

| | | | | |
|-------------|--|---------------|---|---------|
| Syntax | CALibration:CURREnt[:DATA] {<new value>} | | | |
| Description | This command can only be used when calibration is enabled and the output state of the currently selected channel is ON. It enters a current value that is obtained by reading an external meter. The minimum calibration level (CAL:CURREnt:LEV MIN) has to be selected first for the value being entered, then the middle and maximum calibration levels (CAL:CURREnt:LEV MID and CAL:CURREnt:LEV MAX) for the value being entered. Three successive values must be selected and entered. Data values are expressed in base units – either volts or amperes, depending on which function is being calibrated. | | | |
| Parameters | Name | Type | Range | Default |
| | <new value> | numeric (NR2) | -0.2A to MAX + 0.2A The maximum value is dependent on the PSU current rating. See Section 8.1 | – |
| Return | None | | | |
| Usage | See Section 10.2 | | | |

example

Errors 104,"Bad sequence of calibration commands"
 107,"Cal value out of range"

5.1.4. CALibration:CURRent:LEVel

Syntax CALibration:CURRent:LEVel {<level>}

Description This command can only be used when calibration is enabled and the output state of the currently selected channel is ON. It sets the PSU to a calibration point that is entered with the CAL:CURR command. During calibration, three points must be entered and the low-end point (MIN) must be selected and entered first.

This command will set output voltage to MAXimum / 2 (for example 25V for the PSU model with 0-50V).

| Parameters | Name | Type | Range | Default |
|------------------|---|----------|---|---------|
| | <level> | Discrete | MIN MID MAX (see also Section 8.1) | – |
| Return | None | | | |
| Usage example | See Section 10.2 | | | |
| Errors | 101,"Calibration mode is off" 104,"Bad sequence of calibration commands" | | | |
| Related Commands | CALibration:STATe INSTrument:NSElect INSTrument[:SElect] | | | |

5.1.5. CALibration:PASSword:NEW

Syntax CALibration:PASSword:NEW {<old code>, <new code>}

Description Enter a new calibration password. To change the password, first unsecure the PSU using the old password. Then, the new code has to be entered. The calibration code may contain up to 16 characters over the remote interface. Minimum length is 4 characters. The new password is automatically stored in nonvolatile memory and does not have to be stored with CALibrate:SAVE.

| Parameters | Name | Type | Range | Default |
|---------------|---|---------------|--------------------|---------|
| | <old code> | Quoted string | 4 to 16 characters | – |
| | <new code> | Quoted string | 4 to 16 characters | – |
| Return | None | | | |
| Usage example | See Section 10.2 | | | |
| Errors | 102,"Invalid cal password" 105,"Cal password too long" 106,"Cal password too short" | | | |

5.1.6. CALibration:REMark

Syntax CALibration:REMark {<user remark>
 CALibration:REMark?

Description Record calibration information about the PSU. The calibration message is consist of two parts:

- datetime stamp in format *yyyymmdd* and

- up to 32 characters.

The PSU should be in calibration mode before sending a calibration message.

| Parameters | Name | Type | Range | Default |
|------------------|--|---------------|--------------------|----------------------|
| | <user remark> | Quoted string | 0 to 32 characters | "Calibration passed" |
| Return | Query the calibration message. | | | |
| Usage example | When CAL:REM with text "Calibrated by EEZ" is executed at 2015-09-14: CAL:REM "Calibrated by EEZ" CAL:REM? "20150904 Calibrated by EEZ" | | | |
| | See also Section 10.2 | | | |
| Errors | The following errors could be generated by command but not query: 101, "Calibration mode is off" 104, "Bad sequence of calibration commands" | | | |
| Related Commands | CALibration:STATe | | | |

5.1.7. CALibration:SAVE

| | |
|---------------|--|
| Syntax | CALibration:SAVE |
| Description | This command saves calibration constants in non-volatile memory after the calibration procedure has been completed. If calibration mode is exited by programming CALibration:STATe OFF without first saving the new constants, the previous constants are restored. Execution of this command also affects bit 0 (CALibrating) of the Operation Instrument Isummary register (see Section 3.3.2). |
| Return | None |
| Usage example | See Section 10.2 |
| Errors | -340, "Calibration failed" 104, "Bad sequence of calibration commands" 111, "No new cal data exists" |

5.1.8. CALibration:STATe

| | | | | |
|------------------|--|----------|------------|---------|
| Syntax | CALibration:STATe {<bool>} CALibration:STATe? | | | |
| Description | This command enables or disables usage of calibration parameters if they exists. | | | |
| Parameters | Name | Type | Range | Default |
| | <bool> | Discrete | OFF ON 0 1 | ON |
| Return | The returned parameter is 0 (OFF) or 1 (ON). | | | |
| Usage example | CAL:STAT OFF, "eezpsu" | | | |
| Errors | 110, "Cal params missing or corrupted" | | | |
| Related Commands | DIAGnostic[:INFormation]:OTIME? | | | |

5.1.9. CALibration:VOLTage[:DATA]

| | |
|-------------|---|
| Syntax | CALibration:VOLTage[:DATA] {<new value>} |
| Description | This command can only be used when calibration is enabled and the output state of the |

currently selected channel is ON. It enters a voltage value that you obtained by reading an external meter. You must first select the minimum calibration level (CAL:VOLT:LEV MIN) for the value being entered. You must then select the middle and maximum calibration levels (CAL:VOLT:LEV MID and CAL:VOLT:LEV MAX) for the value being entered. Three successive values must be selected and entered. The PSU then computes new voltage calibration constants. These constants has to be stored in non-volatile memory with CALibration:SAVE command.

| Parameters | Name | Type | Range | Default |
|------------------|--|---------------|--|---------|
| | <new value> | numeric (NR2) | -0.5V to MAX + 0.5V The maximum value is dependent on the PSU voltage rating. See Section 8.1 | – |
| Return | None | | | |
| Usage example | See Section Section 10.2 | | | |
| Errors | 104, "Bad sequence of calibration commands" 107, "Cal value out of range" | | | |
| Related Commands | CALibration:SAVE CALibration:STATe INSTrument:NSElect INSTrument[:SElect] | | | |

5.1.10. CALibration:VOLTage:LEVel

Syntax [CALibration:VOLTage:LEVel {<level>}](#)

Description This command can only be used when calibration is enabled and the output state of the currently selected channel is ON. It sets the PSU to a calibration point that is entered with the CAL:VOLT[:DATA] command. During calibration, three points must be entered and the low-end point (MIN) must be selected and entered first.

This command will set output current to 50mA.

| Parameters | Name | Type | Range | Default |
|---------------|---|----------|---|---------|
| | <level> | Discrete | MIN MID MAX (see also Section 8.1) | – |
| Return | None | | | |
| Usage example | See Section Section 10.2 | | | |
| Errors | 101, "Calibration mode is off" 104, "Bad sequence of calibration commands" | | | |

5.2. DIAGnostic

The purpose of the DIAGnostic subsystem is to provide a tree node for all of the PSU service and diagnostic routines used in routine maintenance and repair.

| SCPI command | Description |
|----------------|---|
| DIAGnostic | |
| [:INFormation] | |
| :ADC? | Returns the latest values acquired by ADC |
| :CALibration? | Returns a list of the calibration parameters |
| :FAN? | Returns status of the cooling fan. |
| :PROtection? | Returns the information about all output protections. |
| :TEST? | Returns results of the most recent self-test |

5.2.1. DIAGnostic[:INFormation]:ADC?

| | | | | |
|------------------|--|----------|---------|---------|
| Syntax | DIAGnostic[:INFormation]:ADC? [<channel>] | | | |
| Description | This query returns the latest values acquired by ADC (Analog-to-Digital Converter) of the currently selected channel. | | | |
| Return | Return a list of quoted strings. The U_SET and I_SET are values measure on DAC outputs, and U_MON and I_SET on the channel output binding posts. | | | |
| Parameters | Name | Type | Range | Default |
| | <channel> | Discrete | CH1 CH2 | n/a |
| Usage example | DIAG:ADC? CH2 "U_SET=10.1202", "U_MON=10.12", "I_SET=3.00", "I_MON=1.23" | | | |
| Related Commands | MEASure[:SCALar]:CURRent[:DC] MEASure[:SCALar][:VOLTage][:DC] [SOURce[<n>]]:CURRent[:LEVel][:IMMediate][:AMPLitude] [SOURce[<n>]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] | | | |

5.2.2. DIAGnostic[:INFormation]:FAN?

| | |
|------------------|---|
| Syntax | DIAGnostic[:INFormation]:FAN? |
| Description | Use this query to obtain information about cooling fan state. |
| Return | Returns -1 if cooling fan is not installed (see SYSTem:CPU[:INFormation]?), 0 if fan is stalled (fault condition) or measured fan speed in rpm . |
| Usage example | DIAG:FAN? 1205.00 |
| Related Commands | SYSTem:CPU[:INFormation] |

5.2.3. DIAGnostic[:INFormation]:CALibration?

| | |
|-------------|--|
| Syntax | DIAGnostic[:INFormation]:CALibration? [<channel>] |
| Description | This query returns a list of calibration parameters for the currently selected channel. If the selected channel is in the calibration mode (CALibration[:MODE] ON) then all calibration information collected to the current calibration step will be returned. Otherwise the calibration data stored in non-volatile memory will be returned. |
| Return | The information will be returned as a list of quoted strings. |

| Parameters | Name | Type | Range | Default |
|------------------|--|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | n/a |
| Usage example | <p>Calibration parameters for an 0-40V/0-5A PSU model when the channel 1 is not in the calibration mode:</p> <pre>DIAG:INFO:CAL? CH1</pre> <pre>"remark=20151013 New calibration test", "u_cal_params_exists=1", "u_min_level=0.1V", "u_min_data=0.032V", "u_min_adc=0.06V", "u_mid_level=19.05V", "u_mid_data=18.99V", "u_mid_adc=19.007V", "u_max_level=38.0V", "u_max_data=37.9V", "u_max_adc=37.955V", "i_cal_params_exists=1", "i_min_level=0.01A", "i_min_data=0.005A", "i_min_adc=0.019A", "i_mid_level=2.455A", "i_mid_data=2.45A"; "i_mid_adc=2.464A", "i_max_level=4.9A", "i_max_data=4.82A", "i_max_adc=4.842A"</pre> <p>The query results when a channel is just entered the calibration mode:</p> <pre>DIAG:INFO:CAL?</pre> <pre>"u_level=none", "i_level=none"</pre> <p>The query results when a channel is at the step MIDdle of the voltage calibration:</p> <pre>DIAG:INFO:CAL?</pre> <pre>"u_min=0.11V", "u_level=mid", "u_level_value=24.05V", "u_adc=24.14V", "i_level=none"</pre> | | | |
| Related Commands | <p>CALibration:REMark</p> <p>CALibration:SAVE</p> | | | |

5.2.4. DIAGnostic[:INFormation]:PROTection?

| | |
|------------------|---|
| Syntax | DIAGnostic[:INFormation]:PROTection? |
| Description | This query returns information about all supported output protection mechanisms. |
| Return | The information will be returned as a list of quoted strings. |
| Usage example | <pre>DIAG:PROT?</pre> <pre>"CH1 u_tripped=0", "CH1 u_state=0", "CH1 u_delay=0.10 s", "CH1 i_tripped=0", "CH1 i_state=0", "CH1 i_delay=0.10 s", "CH1 p_tripped=0", "CH1 p_state=1", "CH1 p_delay=5.00 s", "CH1 p_level=50.00 W", "CH2 u_tripped=0", "CH2 u_state=0", "CH2 u_delay=0.10 s", "CH2 i_tripped=0", "CH2 i_state=0", "CH2 i_delay=0.10 s", "CH2 p_tripped=0", "CH2 p_state=0", "CH2 p_delay=5.00 s", "CH2 p_level=50.00 W", "tmain_tripped=0", "tmain_state=1", "tmain_delay=10.00 s", "tmain_level=60.00 oC"</pre> |
| Related Commands | <p>[SOURce[<n>]]:CURRent:PROTection:DElay[:TIME]</p> <p>[SOURce[<n>]]:CURRent:PROTection:STATe</p> <p>[SOURce[<n>]]:CURRent:PROTection:TRIPped?</p> <p>[SOURce[<n>]]:POWEr:PROTection</p> <p>[SOURce[<n>]]:POWEr:PROTection:DElay[:TIME]</p> <p>[SOURce[<n>]]:POWEr:PROTection:STATe</p> <p>[SOURce[<n>]]:POWEr:PROTection:TRIPped?</p> <p>[SOURce[<n>]]:VOLTage:PROTection:DElay[:TIME]</p> <p>[SOURce[<n>]]:VOLTage:PROTection:STATe</p> <p>[SOURce[<n>]]:VOLTage:PROTection:TRIPped?</p> <p>SYSTem:TEMPerature:PROTection[:HIGH][:LEvel]</p> <p>SYSTem:TEMPerature:PROTection[:HIGH]:DElay[:TIME]</p> <p>SYSTem:TEMPerature:PROTection[:HIGH]:STATe</p> <p>SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped?</p> |

5.2.5. DIAGnostic[:INFormation]:TEST?

Syntax [DIAGnostic\[:INFormation\]:TEST?](#)

Description This query returns results of the most recent self-test (see [*TST?](#) command).

Return The information will be returned in the following format: "<return code, device name, installed, return message>" where the return code could be one of the following values:

- 0 – failed
- 1 – passed
- 2 – skipped

This information format will repeat with as many iterations as the number of devices found in configuration parameters of the PSU. While in the Stand-by mode this command will returns only test results for the Arduino shield devices.

Return code for the BP option will always be 2 (skipped).

Usage example Return self-test results where DAC on the channel 2 failed to pass the test and due to that ADC testing has been skipped:

DIAG:TEST?

```
"1, EEPROM, installed, passed", "1, Ethernet, installed, passed"
"2, RTC, not installed, skipped", "2, BP option, installed,
skipped", "1, FAN, installed, passed", "1, CH1 IOEXP, installed,
passed", "1, CH1 DAC, installed, passed", "1, CH1 ADC, installed,
passed", "1, CH2 IOEXP, installed, passed", "0, CH2 DAC,
installed, failed", "1, CH2 ADC, installed, skipped"
```

Return self-test results while PSU is in the Stand-by mode:

DIAG:TEST?

```
"1, EEPROM, installed, passed", "1, Ethernet, installed, passed"
"1, RTC, installed, passed", "2, BP option, installed, skipped",
"1, FAN, installed, passed"
```

Related [*TST?](#)

Commands SYSTem:POWer

5.3. INSTRument

Each channel of the PSU is considered as separate (logical) instrument, which is required by the SCPI standard. The INSTRument subsystem provides a mechanism to identify and select instruments and establish coupling to simplify programming of more channels at once.

| SCPI command | Description |
|--|-------------------------------------|
| INSTRument | |
| <code>[[:SElect] {<channel>}</code> | Selects the output to be programmed |
| <code>:NSElect {<channel_number>}</code> | Selects the output to be programmed |

5.3.1. INSTRument[:SElect]

Syntax `INSTRument[:SElect] {<channel>}`
`INSTRument[:SElect]?`

Description This command selects the output to be programmed by the output identifier. The outputs of the PSU are considered as two logical instruments. The INSTRument command provides a mechanism to identify and select an output.
When one output is selected, the other output is unavailable for programming until selected. The following commands are affected by the INSTRument command: SOURce, MEASure, and CALibration.

| Parameters | Name | Type | Range | Default |
|------------------|---|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |
| Return | Query returns the currently selected output by the INSTRument[:SElect] or INSTRument:NSElect command. The returned value is CH1 or CH2. | | | |
| Usage example | INST? CH1 INST:SEL? CH1 | | | |
| Related Commands | INSTRument:NSElect | | | |

5.3.2. INSTRument:NSElect

Syntax `INSTRument:NSElect {<channel_number>}`
`INSTRument:NSElect?`

Description This command is used in conjunction with the SElect command. It serves the same purpose, except that it uses a numeric value instead of the identifier used in the SElect command.

| Parameters | Name | Type | Range | Default |
|------------------|--|---------|-------|---------|
| | <channel_number> | Numeric | 1 2 | – |
| Return | When queried it returns the logical instrument number of the currently selected PSU channel. Note that the numbering used for logical instruments directly corresponds to the numbers used in status reporting for multiple instruments; specifically the STATUS:QUESTIONable:INSTRument and STATUS:OPERation:INSTRument commands. | | | |
| Usage example | INST:NSEL 2 INST:NSEL? 2 | | | |
| Related Commands | STATUS:QUESTIONable:INSTRument STATUS:OPERation:INSTRument | | | |

5.4. MEASure

Measure commands measure the output voltage, current, power or temperature. They trigger the acquisition of new data before returning the reading. Measurements are performed by digitizing the instantaneous output voltage, current or temperature. Output power is calculated as product of measured voltage and current. Keyword [:DC] is optional since all measurement are by default of the DC level of the signal.

| SCPI command | Description |
|--------------------|--|
| MEASure | |
| [:SCALar] | |
| :CURRent | |
| [:DC]? [<channel>] | Takes a measurement; returns the average current |
| :POWer | |
| [:DC]? [<channel>] | Takes a measurement; returns the average power |
| :TEMPerature | |
| [:DC]? {<sensor>} | Takes a measurement; returns the average temperature |
| [:VOLTage] | |
| [:DC]? [<channel>] | Takes a measurement; returns the average voltage |

5.4.1. MEASure[:SCALar]:CURRent[:DC]

Syntax [MEASure\[:SCALar\]:CURRent\[:DC\]? \[<channel>\]](#)

Description Query the current measured across the current sense resistor inside the PSU.

| Parameters | Name | Type | Range | Default |
|------------------|--|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |
| Return | Returns the average output current in amperes as decimal number (NR2). | | | |
| Usage example | Measure current on the currently selected channel (CH1) and CH2: MEAS:CURR?; :MEAS:CURR? CH2 1.23;0.12 | | | |
| Related Commands | | | | |

5.4.2. MEASure[:SCALar]:POWer[:DC]

Syntax [MEASure\[:SCALar\]:POWer\[:DC\]? \[<channel>\]](#)

Description Query the output power calculated as product of measured voltage and current value.

| Parameters | Name | Type | Range | Default |
|------------------|--|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | n/a |
| Return | Returns the average output power in watts as decimal number (NR2). | | | |
| Usage example | MEAS:POW? CH2 80.44 | | | |
| Related Commands | | | | |

5.4.3. MEASure[:SCALar]:TEMPerature[:THERmistor][:DC]

Syntax [MEASure\[:SCALar\]:TEMPerature\[:THERmistor\]\[:DC\]? {<sensor>}](#)

Description Query the temperature measured using the specified temperature sensors as follows:

MAIN – Analog temperature sensor connected to Arduino Shield board (can be used to monitor i.e. the main heatsink temperature or power transformer temperature).

Return Returns the average temperature value in degrees Celsius (°C) as decimal number (NR2).

| Parameters | Name | Type | Range | Default |
|------------|----------|----------|--------|---------|
| | <sensor> | Discrete | [MAIN] | MAIN |

Usage example
MEAS:TEMP? MAIN
39.50

Errors

Related
Commands

5.4.4. MEASure[:SCALar][:VOLTage][:DC]

Syntax [MEASure\[:SCALar\]\[:VOLTage\]\[:DC\]? \[<channel>\]](#)

Description Query the voltage measured at the sense terminals of the selected channel.

| Parameters | Name | Type | Range | Default |
|------------|-----------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | n/a |

Return Returns the average output voltage in volts as decimal number (NR2).

Usage example
MEAS:VOLT? CH1
43.25

Query voltage of the channel 2 that is currently selected:

```
INST CH2
MEAS?
12.40
```

Related
Commands

5.5. MEMory

The MEMory subsystem works with PSU state files that are saved to ([*SAV](#)) and recalled from ([*RCL](#)) non-volatile storage locations numbered 0 through 9. The storage location 0 named "Power down state" is used to store the current PSU parameters.

| SCPI command | Description |
|---|--|
| MEMory | |
| :NSTates? | Returns total number of state storage memory locations |
| :STATe | |
| :CATalog? | Lists the names associated with all ten state storage locations |
| :DElete {<profile>} | Deletes the contents of a state storage location |
| :ALL | Deletes the contents of all state storage locations |
| :NAME {<profile>, <name>} | Assigns a custom name to a state storage locations |
| :RECall | |
| :AUTO {<bool>} | Specifies whether the power-down state is recalled from location 0 on power-on |
| :SElect {<profile>} | Specifies which PSU state will be used at power on |
| :VALid? {<profile>} | Determines whether a storage location contains a valid state |

5.5.1. MEMory:NSTates

| | |
|------------------|--|
| Syntax | MEMory:NSTates? |
| Description | Returns the total number of *SAV/*RCL states available in the PSU. |
| Return | Returns numeric value which is one greater than the maximum that can be sent as a parameter to the *SAV and *RCL commands. |
| Usage example | MEM:NST? 10 |
| Errors | |
| Related Commands | |

5.5.2. MEMory:STATe:CATalog

| | |
|------------------|---|
| Syntax | MEMory:STATe:CATalog? |
| Description | This query requests a list of defined names in the MEMory:STATe subsystem. |
| Return | The PSU returns a list of defined <name>'s in a comma separated list. Each <name> is returned in a quoted string. |
| Usage example | MEM:STAT:CAT? "Power down state", "All outputs on", "dual 15V/300mA", "Power protection at 100W", "--Not used--", "--Not used--", "--Not used--", "--Not used--", "--Not used--", "--Not used--" |
| Errors | |
| Related Commands | MEMory:STATe:NAME |

5.5.3. MEMory:STATe:DElete

| | |
|--------|--|
| Syntax | MEMory:STATe:DElete {<profile>} MEMory:STATe:DElete:ALL |
|--------|--|

Description When used with a profile number this command deletes the contents of the specified storage location. The MEMory:STATe:DELeTe:ALL deletes the contents of storage locations 1 through 9.

An error is generated on an attempt to recall a deleted state.

| Parameters | Name | Type | Range | Default |
|-------------------------|----------------|---------|-------|---------|
| | <profile> | Numeric | 1 – 9 | – |
| Return | None | | | |
| Usage example | MEM:STAT:DEL 2 | | | |
| Errors | | | | |
| Related Commands | *RCL *SAV | | | |

5.5.4. MEMory:STATe:NAME

Syntax MEMory:STATe:NAME {<profile>, <name>}
MEMory:STATe:NAME? {<profile>}

Description This command associates a <name> with a *SAV/*RCL register number. May assign same name to different locations and state names are unaffected by [*RST](#). Deleting a storage location's contents MEMory:STATe:DELeTe resets associated name to "--Not used--"

| Parameters | Name | Type | Range | Default |
|-------------------------|--|---------------|--------------------|---------|
| | <profile> | Numeric | 1 – 9 | – |
| | <name> | quoted string | 0 to 32 characters | – |
| Return | Returns a *SAV/*RCL register number associates with profile number. | | | |
| Usage example | MEM:STAT:DEF, 2, "All outputs on" MEM:STAT:DEF? 2 "All outputs on" | | | |
| Errors | | | | |
| Related Commands | MEMory:STATe:DELeTe | | | |

5.5.5. MEMory:STATe:RECall:AUTO

Syntax MEMory:STATe:RECall:AUTO {<bool>}
MEMory:STATe:RECall:AUTO?

Description This command disables or enables the automatic recall of a specific stored PSU state selected using the MEMory:STATe:RECall:SELeCt command when power is turned on. Select ON to automatically recall one of the ten stored states or the "power-down" state (location 0) when power is turned on. Select OFF to issue a reset ([*RST](#)) when power is turned on.

| Parameters | Name | Type | Range | Default |
|----------------------|--|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | – |
| Return | The query command returns 0 (OFF) or 1 (ON). | | | |
| Usage example | MEM:STAT:REC:AUTO? 1 | | | |
| Errors | | | | |
| Related | *SAV | | | |

Commands MEMory:STATe:RECall:SElect
SYSTem:POWer

5.5.6. MEMory:STATe:RECall:SElect

Syntax MEMory:STATe:RECall:SElect {<profile>}
MEMory:STATe:RECall:SElect?

Description This command selects which PSU state will be used at power on if the automatic recall mode is enabled (see MEMory:STATe:RECall:AUTO ON command). If the automatic recall mode is disabled (MEMory:STATe:RECall:AUTO OFF), then a “factory reset” (return to the default values) is issued when power is turned on.

| Parameters | Name | Type | Range | Default |
|------------|-----------|---------|-------|---------|
| | <profile> | Numeric | 0 – 9 | – |

Return The query command returns numeric value for 0 to 9 indicating which instrument state will be used at power on.

Usage example MEM:STAT:REC:SEL?
2

Errors

Related *SAV
Commands MEMory:STATe:RECall:AUTO
SYSTem:POWer

5.5.7. MEMory:STATe:VALid

Syntax MEMory:STATe:VALid? {<profile>}

Description This command queries the specified storage location to determine if a valid state is currently stored in this location.

*Use this command before sending the [*RCL](#) command to determine if a valid state has been previously stored on queried location.*

| Parameters | Name | Type | Range | Default |
|------------|-----------|----------|-------|---------|
| | <profile> | Discrete | 0 – 9 | – |

Return Returns 0 if no state has been stored or if it has been deleted. It returns 1 if a valid state is stored in this location.

Usage example MEM:STAT:VAL? 2
1

Errors

Related *RCL
Commands *SAV

5.6. OUTPut

The OUTPut subsystem controls the output, power-on, protection reset and tracking state.

| SCPI command | Description |
|---|--|
| OUTPut | |
| [:STATe] {<bool>} | Controls the specified channel output state |
| :DPRog {<bool>} | Controls down-programmer circuit |
| :MODE? | Returns the channel mode of operation |
| :PROTection | |
| :CLEar | Resets latched protection |
| :SENSe | Enables the remote sense function of the channel (obsolete, see [SOURce[<n>]]:VOLTage:SENSe[:SOURce]) |

5.6.1. OUTPut[:STATe]

Syntax [OUTPut\[:STATe\] {<bool>} \[, <channel>\]](#)
[OUTPut\[:STATe\]? \[<channel>\]](#)

Description This command enables or disables the specified output channel(s). The enabled state is ON (1); the disabled state is OFF (0). The state of a disabled output is a condition of zero output voltage and zero source current. Execution of this command also affects bit 10 (OE) and bit 11 (DP) of the Operation Instrument Summary register (see [Section 3.3.2](#)). Self-test operation initiated by [*TST?](#) command will put all PSU channels into disable state. When the BP_OPTION is enabled and the channels are not grouped together (SYSTem:GROUp commands scheduled for M3) this command sets LED indicators above binding posts in the following manner:

- Turn on/off Out1+/Out1- green indicators (LED_O1+, LED_O1- or TLC5925 Out7 and Out3) when CH1 is selected
- Turn on/off Out2+/Out2- indicators (LED_O2+, LED_O2- or TLC5925 Out13 and Out10) when CH2 is selected

Execution of the OUTP ON command on the channel with tripped one or more protection (OCP, OVP, OPP or OTP) will generate error 201. Use OUTPut:PROTection:CLEar command to clear all tripped protections.

This command also affects Sense+/Sense- indicators (LED_S1+, LED_S1-, LED_S2+, LED_S2-) controlled by the [\[SOURce:\]VOLTage:SENSe\[:SOURce\]](#) command.

| Parameters | Name | Type | Range | Default |
|------------------|--|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | — |
| | <channel> | Discrete | CH1 CH2 | — |
| Return | The query command returns 0 if the output is OFF, and 1 if the output is ON. | | | |
| Usage example | OUTP ON, CH1 OUTP? CH1 1 | | | |
| Errors | 108, "Cal output disabled" 201, "Cannot execute before clearing protection" | | | |
| Related Commands | *TST OUTPut:PROTection:CLEar [SOURce:]VOLTage:SENSe[:SOURce] | | | |

5.6.2. OUTPut:DPRog

Syntax **OUTPut:DPRog** {<bool>}
OUTPut:DPRog?

Description A down-programmer is a circuit built into the channel's output of a PSU that actively pulls the output voltage down when the PSU's channel is moving from a higher setting to a lower setting.
 The down-programmer circuit is active by default and only rare situation requires to be turned off. One such situation is when battery is connected as a load. Another one is connecting two channel in parallel when only one down-programmer circuit is enough for actively pulls the output voltage down while lower voltage is set.
 Execution of this command also affects bit 11 (DP) of the Operation Instrument Summary register (see Section [Section 3.3.2](#)).

Despite of the down-programmer state programmed by this command, it will be deactivated when the channel output is turned off (i.e. OUTPut[:STATe] OFF or). When the channel output is turned on again, down-programmer will be set back to the state programmed with this command.

| Parameters | Name | Type | Range | Default |
|------------------|----------------|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | ON |
| Usage example | OUTP:DPR? 1 | | | |
| Related Commands | OUTPut[:STATe] | | | |

5.6.3. OUTPut:MODE?

Syntax **OUTPut:MODE?** [, <channel>]

Description This command simplify resolving a results that can be obtained reading the bit 8 (CV) and 9 (CC) of the read-only Instrument Summary Operation Status register for a specific channel (see table in the [Section 3.3.2](#)). The PSU can works in one of the three output modes:

- CV (Constant Voltage), when the output voltage equals the voltage setting value and the output current is determined by the load
- CC (Constant Current), when the output current equals the current setting value and the output voltage is determined by the load and
- UR (Unregulated) that is critical mode between CV and CC modes that could be noticed when the output voltage is close to the one end of the full scale (i.e. somewhere below 2V and above 48V for the 50V model that depends of the connected load).

The UR mode is not supported in [software simulator](#).

| Parameters | Name | Type | Range | Default |
|---------------|--|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |
| Return | The query returns CV, CC or UR. | | | |
| Usage example | Set output voltage to 20V and max. current, check that output voltage is as defined that indicate the constant voltage mode of operation: VOLT 20; CURR MAX MEAS:VOLT? 20.0 OUTP:MODE? CV | | | |

Errors

Related STATus:OPERation:INSTrument:ISUMmary[<n>][:EVENT]

Commands STATus:OPERation:INSTrument:ISUMmary[<n>]:CONDition

 STATus:OPERation:INSTrument:ISUMmary[<n>]:ENABLE

5.6.4. OUTPut:PROTection:CLEar

Syntax [OUTPut:PROTection:CLEar \[<channel>\]](#)

Description This command clears the latched protection status that disables the output when an over-voltage, over-current or a power-limit condition is detected. All conditions that generate the fault must be removed before the latched status can be cleared. The output is restored to the state it was in before the fault condition occurred.

| Parameters | Name | Type | Range | Default |
|------------|-----------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |

Return None

Usage example The following command clears the latched protection status on all channels:
OUTP:PROT:CLE

Related
Commands

5.6.5.

5.7. SOURce

The SOURce commands are used to set the output voltage and current values, remote voltage sensing, and implemented protection mechanisms on the specified channel. Although the [APPLY](#) command provides the most straightforward method to program the PSU over the remote interfaces, the SOURce commands give you more flexibility to change individual parameters.

| SCPI command | Description |
|--------------------------------------|--|
| [SOURce[<n>]] | |
| :CURRent | |
| [:LEVel] | |
| [:IMMediate][:AMPLitude] {<current>} | Sets the output current |
| :STEP[:INCRement] {<step>} | Sets the step of the current change |
| :PROTectioN | |
| :DELay | |
| [:TIME] {<time>} | Sets the over-current protection (OCP) programming delay |
| :STATe {<bool>} | Enables/disables over-current protection on the selected channel |
| :TRIPped? | Returns status of over-current protection activation |
| :LRIPple {<bool>} | Sets low output ripple (noise) mode of operation |
| :AUTO {<bool>} | Sets automatic changing to the low output ripple (noise) mode of operation |
| :POWER | |
| :PROTectioN[:LEVel] | Sets the over-power protection (OPP) level |
| :DELay | |
| [:TIME] {<time>} | Sets the over-power protection programming delay |
| :STATe {<bool>} | Enables/disables over-power protection on the selected channel |
| :TRIPped? | Returns status of over-power protection activation |
| :VOLTage | |
| [:LEVel] | |
| [:IMMediate][:AMPLitude] {<voltage>} | Sets the output voltage |
| :STEP[:INCRement] {<step>} | Sets the step of the voltage change |
| :PROGram[:SOURce] {<source>} | Sets voltage programming source |
| :PROTectioN | |
| :DELay | |
| [:TIME] {<time>} | Sets the over-voltage protection (OVP) programming delay |
| :STATe {<bool>} | Enables/disables over-voltage protection on the selected channel |
| :TRIPped? | Returns status of over-voltage protection activation |
| :SENSe[:SOURce] {<source>} | Sets voltage sense inputs source |

5.7.1. [SOURce[<n>]]:CURRent

| | | | | |
|------------------|--|-------------------------|---|---------|
| Syntax | [SOURce[<n>]]:CURRent[:LEVel][:IMMediate][:AMPLitude] {<current> MIN DEF MAX UP DOWN} [SOURce[<n>]]:CURRent[:LEVel][:IMMediate][:AMPLitude]? [MIN DEF MAX] | | | |
| Description | <p>This command sets the immediate current level of the channel. Units are in amperes. When [:SOURce[<n>]] or [<n>] is omitted, the currently selected channel will be affected by this command.</p> <p>This command also increases or decreases the immediate current level using the 'UP' or DOWN parameter by a predetermined amount. The command CURRent:STEP sets the amount of increase or decrease. A new increment setting will <i>not</i> cause an execution error -222,"Data out of range" when the maximum or the minimum rated current is exceeded – the output value will be set to the maximum or the minimum value instead.</p> <p>At *RST, the signal being sourced will be set to a "safe" condition. This is achieved by setting the amplitude to its MINimum value (see Section 8.1).</p> | | | |
| Return | The query command returns the programmed current level. CURR? MIN, CURR? DEF and CURR? MAX can be used to obtain minimum, default and maximum current level on the currently selected channel. For actual output current value use MEASure:CURRent command. | | | |
| Parameters | Name | Type | Range | Default |
| | <current> | Numeric (NR2), discrete | 0 to MAXimum, MIN DEF MAX UP DOWN The maximum value is dependent on the PSU current rating. See Section 8.1 | – |
| Usage example | <p>A 10 ohm load is connected and voltage is set to 20V. With MAX current set measured current will be 2A. When new current value is set to 1.2A, voltage will drop to 12V (the channel enters the CC mode of operation):</p> <pre> INST CH1 VOLT 20 CURR MAX MEAS:VOLT? 20.00 CURR 1.2 MEAS:VOLT? 12.00 Query that returns maximum current of the currently selected channel: CURR? MAX 5.00 </pre> | | | |
| Errors | 150,"Power limit exceeded" -222,"Data out of range" | | | |
| Related Commands | *RST MEASure[:SCALar]:CURRent[:DC]? [SOURce[<n>]]:CURRent[:LEVel][:IMMediate]:STEP[:INCRement] | | | |

5.7.2. [SOURce[<n>]]:CURRent:STEP

| | | | | |
|-------------|---|--|--|--|
| Syntax | [SOURce[<n>]]:CURRent[:LEVel][:IMMediate]:STEP[:INCRement] {<step> DEFAult} [SOURce[<n>]]:CURRent[:LEVel][:IMMediate]:STEP[:INCRement]? [DEFAult] | | | |
| Description | Set the step of the current change of the channel. When [SOURce[<n>]] or [<n>] is omitted, the currently selected channel will be affected by this command. Step change is per- | | | |

formed by using UP and DOWN as parameter for the [SOURce[<n>]]:CURRent command.

Return The query returns the step of the current change of the specified channel.

| Parameters | Name | Type | Range | Default |
|------------|------|------|-------|---------|
|------------|------|------|-------|---------|

| | | | | |
|--|--------|---------------|--------------------|------|
| | <step> | Numeric (NR2) | 0.01 to 1A DEFault | 0.05 |
|--|--------|---------------|--------------------|------|

Usage example Return default step value:

```
CURR:STEP? DEF
```

```
0.05
```

When a 10 ohm load is connected with voltage set to 20V and current to 1A the first channel enters CC mode of operation. Current is then increased from 1A in two steps to 1.2A:

```
APPL CH1, 20, 1
```

```
MEAS:VOLT?
```

```
10.00
```

```
CURR:STEP 0.1
```

```
CURR UP
```

```
MEAS:CURR?
```

```
1.10
```

```
CURR UP
```

```
MEAS:CURR?
```

```
1.20
```

```
MEAS:VOLT?
```

```
12.00
```

Errors

Related Commands [SOURce[<n>]]:CURRent

5.7.3. [SOURce[<n>]]:CURRent:PROTectio:n:DELay[:TIME]

Syntax [SOURce[<n>]]:CURRent:PROTectio:n:DELay[:TIME] {<time>|DEFault}
[SOURce[<n>]]:CURRent:PROTectio:n:DELay[:TIME]? [DEFault]

Description This command sets the over-current protection delay. The over-current protection function will not be triggered on the selected output channel during the delay time. After the delay time has expired, the over-current protection function will be active. This prevents momentary changes in output status from triggering the over-current protection function. Programmed values can range from 0 to 10 seconds. See also [Section 8.1](#)

Return The query command returns the programmed delay time.

| Parameters | Name | Type | Range | Default |
|------------|------|------|-------|---------|
|------------|------|------|-------|---------|

| | | | | |
|--|--------|-------------------|----------------|------|
| | <time> | Numeric, discrete | 0 – 10 DEFault | 20ms |
|--|--------|-------------------|----------------|------|

Usage example Get default OCP delay of 20 milliseconds:

```
CURR:PROT:DEL? DEF
```

```
0.02
```

Errors

Related Commands OUTPut:PROTectio:n:CLEar

5.7.4. [SOURce[<n>]]:CURRENT:PROTection:STATe

Syntax [SOURce[<n>]]:CURRENT:PROTection:STATe {<bool>}
[SOURce[<n>]]:CURRENT:PROTection:STATe?

Description This command enables or disables the over-current protection (OCP) function. The enabled state is ON (1); the disabled state is OFF (0).
Since the PSU do not have a dedicated over-current protection circuit that can be programmed independently of output current level, entering the CC (constant current) mode of operation is used as a trigger to start OCP sequence. When delay time specified with the [SOURce[<n>]]:CURRENT:PROTection:DELaY[:TIME] command expired the output turns off and the Questionable Condition status register OCP bit 9 is set. An error tone will also follow if beeper is enabled (see SYSTem:BEEPer:STATe).
[SOURce[<n>]]:CURRENT:PROTection:TRIPped? command can be used to query whether over-current protection occurred on the selected channel.

Return The query command returns 0 if the current protection state is OFF, and 1 if the current protection state is ON.

| Parameters | Name | Type | Range | Default |
|------------|--------|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |

Usage example
CURR:PROT:STAT?
0

Errors

Related OUTPut:PROTection:CLEAr
Commands [SOURce[<n>]]:CURRENT:PROTection:DELaY[:TIME]
[SOURce[<n>]]:CURRENT:PROTection:TRIPped
SYSTem:BEEPer:STATe

5.7.5. [SOURce[<n>]]:CURRENT:PROTection:TRIPped?

Syntax [SOURce[<n>]]:CURRENT:PROTection:TRIPped?

Description Query whether OCP occurred on the currently selected channel. When protection is tripped bit 9 (OCP) of the Questionable Instrument Isummary register will be set (see [Section 3.4.2](#)).
The [OUTPut:PROTection:CLEAr](#) command can be send to clear OCP condition on the selected channel.

Return This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

Usage example
CURR:PROT:TRIP?
1

Errors

Related OUTPut:PROTection:CLEAr
Commands

5.7.6. [SOURce[<n>]]:LRIPple

Syntax [SOURce[<n>]]:LRIPple {<bool>}
[SOURce[<n>]]:LRIPple?

Description This command enables or disables low output ripple (noise) mode of operation when it's supported by installed channel board (use the SYSTem:CHANnel[:INfOrmation]:PROGram? to query channel functionality).
When low ripple mode of operation is selected the power pre-regulator is bypassed by setting so-called [Duty cycle](#) of the SMPS controller to 100 %. That will disable switching frequency and therefore that otherwise hard to filter component disappear entirely from the channel output. In that case the remaining noise mainly comes from the low power

bias switching pre-regulator.

The maximum output power in this mode of operation is limited to stay within [SOA](#) (Safe operating area) of the pre-regulator and post-regulator regulation elements. The maximum output power is limited by the first of the following conditions that is met:

- The pre-regulator regulation element (switching mosfet) capability is limited with max. allowed continuous current (*SOA_PREG_CURR* value) for the applied input voltage (*SOA_VIN* value). The set output current cannot exceeds this value in any moment.
- The post-regulator regulation element (pass mosfet) could dissipate finite power (*SOA_POSTREG_PTOT* value) while load is connected. Therefore output power cannot exceed product of voltage difference (*SOA_VIN* and output voltage) and output current.

For example, when output voltage is set to 12 V and *SOA_VIN* = 52 V, *SOA_PREG_CURR* = 1 A and *SOA_POSTREG_PTOT* = 20 W, the max. output current in low ripple mode cannot exceed 0.5 A because *SOA_VIN-VOLT* = 52-12 = 40 V and *I*_{max}=*SOA_POSTREG_PTOT*/40 = 20/40 = 0.5 A.

If output voltage is set to 36 V with the same SOA values as in example above the max. output current cannot exceed 1 A (limited by *SOA_PREG_CURR*) regardless of the fact that max. power dissipation of 20 W allows output current of up to 1.25 A or 20/(52-16).

If low ripple mode is set an error will be generated when any of the limits are exceeded. The AUTOMATIC mode should be used to avoid error conditions. Also if AUTOMATIC mode disables low ripple, an attempt to enable it using the [SOURCE<n>]:LRIPple ON command will generate an execution error.

| Parameters | Name | Type | Range | Default |
|------------------|---|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| Return | The query command returns 1 if the low ripple mode is active, and 0 if the low ripple mode is not active. | | | |
| Usage example | VOLT:LRIP? 0 | | | |
| Errors | -200, "Execution error" 302, "Option not installed" | | | |
| Related Commands | [SOURCE<n>]:LRIPple:AUTO SYSTem:CHANnel:INFormation:PROGram? | | | |

5.7.7. [SOURCE<n>]:LRIPple:AUTO

| | | | | |
|-------------|---|--|--|--|
| Syntax | [SOURCE<n>]:LRIPple:AUTO {<bool>} [SOURCE<n>]:LRIPple:AUTO? | | | |
| Description | Use this command to allows automatic changing to the low ripple (noise) mode of operation. If AUTOMATIC mode of operation is enabled, the channel will stay in low ripple mode of operation as long as output power does not exceed calculated limitations allowed for that mode of operation. The [SOURCE<n>]:LRIPple? query can be used to determine current status. | | | |

| Parameters | Name | Type | Range | Default |
|---------------|---|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| Return | The query command returns 1 if the low ripple automatic mode is active, and 0 if the low ripple automatic mode is not active. | | | |
| Usage example | VOLT:LRIP:AUTO 1 VOLT:LRIP? 1 | | | |

Errors 302, "Option not installed"

Related SOURce[<n>]:LRIPple

Commands SYSTem:CHANnel:INFOrmation:PROGram?

5.7.8. [SOURce[<n>]]:POWer:PROTection[:LEVel]

Syntax [SOURce[<n>]]:POWer:PROTection[:LEVel] {<current>|MINimum|DEFault|MAXimum}
[SOURce[<n>]]:POWer:PROTection[:LEVel]?

Description Set the over-power protection (OPP) value of the channel. When [:SOURce[<n>]] or [<n>] is omitted, the currently selected channel will be affected by this command. When the over-power protection function of the specified channel is enabled ([SOURce[<n>]]:POWer:PROTection:STATe), the output turns off automatically when the output power exceeds the over-power protection value currently set. [SOURce[<n>]]:POWer:PROTection:TRIPped? command can be used to query whether over-power protection occurred on the selected channel.

Return Query the over-power protection (OPP) value of the selected channel.

| Parameters | Name | Type | Range | Default |
|------------|-----------|-------------------------|---|---------|
| | <current> | Numeric (NR2), discrete | 0 to maximum, MIN DEF MAX The maximum value is dependent on the PSU power rating. See Section 8.1 | DEFault |

Usage Set power protection to 50W on the channel 2:

example SOUR2:POW:PROT 50

Errors

Related [SOURce[<n>]]:POWer:PROTection:TRIPped?

Commands [SOURce[<n>]]:POWer:PROTection:STATe

5.7.9. [SOURce[<n>]]:POWer:PROTection:DELay[:TIME]

Syntax [SOURce[<n>]]:POWer:PROTection:DELay[:TIME] {<time>|DEFault}
[SOURce[<n>]]:POWer:PROTection:DELay[:TIME]? [DEFault]

Description This command sets the over-power protection (OPP) delay. The over-power protection function will not be triggered on the selected output channel during the delay time. After the delay time has expired, the over-power protection function will be active. This prevents momentary changes in output status from triggering the over-power protection function. Programmed values can range from 0 to 300 seconds. See also [Section 8.1](#)

Return The query command returns the programmed delay time.

| Parameters | Name | Type | Range | Default |
|------------|--------|---------|-----------------|---------|
| | <time> | Numeric | 0 – 300 DEFault | 10 |

Usage Get default OPP delay of 10 seconds:

example POW:PROT:DEL? DEF
10

Errors

Related OUTPut:PROTection:CLEAr

Commands

5.7.10. [SOURce[<n>]]:POWer:PROTection:STATe

| | | | | |
|------------------|---|----------|------------|---------|
| Syntax | [SOURce[<n>]]:POWer:PROTection:STATe {<bool>} [SOURce[<n>]]:POWer:PROTection:STATe? | | | |
| Description | This command enables or disables the over-power protection (OPP) function. The enabled state is ON (1); the disabled state is OFF (0). If the over-power protection function is enabled and the measure output power reach value set by [SOURce[<n>]]:POWer:PROTection[:LEVel] the output is disabled and the Questionable Condition status register OPP bit 10 is set. | | | |
| Return | The query command returns 0 if the current protection state is OFF, and 1 if the current protection state is ON. | | | |
| Parameters | Name | Type | Range | Default |
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| Usage example | POW:PROT:STAT ON POW:PROT:STAT? 1 | | | |
| Errors | | | | |
| Related Commands | OUTPut:PROTection:CLEAr [SOURce[<n>]]:POWer:PROTection[:LEVel] | | | |

5.7.11. [SOURce[<n>]]:POWer:PROTection:TRIPped?

| | | | | |
|------------------|---|--|--|--|
| Syntax | [SOURce[<n>]]:POWer:PROTection:TRIPped? | | | |
| Description | Query whether OPP occurred on the currently selected channel. When protection is tripped bit 10 (OPP) of the Questionable Instrument Isummary register will be set (see Section 3.4.2). The OUTPut:PROTection:CLEAr command can be send to clear OPP condition on the selected channel. | | | |
| Return | This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped. | | | |
| Usage example | POW:PROT:TRIP? 0 | | | |
| Errors | | | | |
| Related Commands | OUTPut:PROTection:CLEAr | | | |

5.7.12. [SOURce[<n>]]:VOLTage

| | | | | |
|-------------|---|--|--|--|
| Syntax | [SOURce[<n>]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] {<voltage> MINimum DE-Fault MAXimum UP DOWN} [SOURce[<n>]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]? [MINimum DEFault MAXimum] | | | |
| Description | This command sets the immediate voltage level of the output channel. Units are in volts. When [:SOURce[<n>]] or [<n>] is omitted, the currently selected channel will be affected by this command. This command also increases or decreases the immediate voltage level using the 'UP' or DOWN parameter by a predetermined amount. The command VOLTage:STEP sets the amount of increase or decrease. A new increment setting will <i>not</i> cause an execution error -222,"Data out of range" when the maximum or the minimum rated current is exceeded – the output value will be set to the maximum or the minimum value instead. At *RST, the signal being sourced will be set to a "safe" condition. This is achieved by setting the amplitude to its MINimum value (see Section 8.1). | | | |

Return The query command returns the programmed voltage level. VOLT? MIN, VOLT? DEF and VOLT? MAX can be used to obtain minimum, default and maximum voltage level on the currently selected channel. For actual output voltage value use MEASure:VOLTage? command.

| Parameters | Name | Type | Range | Default |
|------------|-----------|-------------------------|---|---------|
| | <voltage> | Numeric (NR2), discrete | 0 to MAXimum, MIN DEF MAX UP DOWN The maximum value is dependent on the PSU voltage rating. See Section 8.1 | – |

Usage example A 10 ohm load is connected and current is set to 1A. With MAX voltage set measured voltage will be 10V. When new voltage value is set to 5V, current will drop to 0.5A (the channel enters the CV mode of operation):

```
INST CH1
VOLT MAX
CURR 1
MEAS:CURRE?
```

```
1.00
```

```
VOLT 5
MEAS:CURRE?
```

```
0.50
```

Query that returns maximum current of the currently selected channel:

```
CURRE? MAX
```

```
5.00
```

Errors 150, "Power limit exceeded"
-222, "Data out of range"

Related *RST

Commands MEASure[:SCALar]:VOLTage[:DC]?
[SOURce[<n>]]:VOLTage[:LEVel][:IMMediate]:STEP[:INCRement]

5.7.13. [SOURce[<n>]]:VOLTage:STEP

Syntax [SOURce[<n>]]:VOLTage[:LEVel][:IMMediate]:STEP[:INCRement] {<step>|DEFault}
[SOURce[<n>]]:VOLTage[:LEVel][:IMMediate]:STEP[:INCRement]? [DEFault]

Description Set the step of the voltage change of the channel. When [SOURce[<n>]] or [<n>] is omitted, the currently selected channel will be affected by this command. Step change is performed by using UP and DOWN as parameter for the [SOURce[<n>]]:VOLTage command.

Return The query returns the step of the voltage change of the specified channel.

| Parameters | Name | Type | Range | Default |
|------------|--------|-------------------------|---------------------|---------|
| | <step> | Numeric (NR2), discrete | 0.01 to 10V DEFault | 0.1 |

Usage example Return default step value:

```
VOLT:STEP? DEF
```

```
0.10
```

By connecting a 10 ohm load and current set to 2A and voltage to 10V the first channel enters the CV mode of operation. Voltage is then decreased from 10V in two steps to 6V:

```
APPL CH1, 10, 2
MEAS:CURRE?
```

```
1.0
```

```

VOLT:STEP 2
VOLT DOWN
VOLT DOWN
MEAS:VOLT?

6.0

MEAS:CURRE?
0.60

```

Errors

Related [SOURce[<n>]]:VOLTage
Commands

5.7.14. [SOURce[<n>]]:VOLTage:PROGram[:SOURce]

Syntax [SOURce:]VOLTage:PROGram[:SOURce] {INTernal|EXTernal}
[SOURce:]VOLTage:PROGram[:SOURce]?

Description Use this command to define signal source for output voltage programming if channel support this option (use the 5.9.7. SYSTem:CHANnel[:INFormation]:PROGram? to query channel programming capability functionality).
A channel's D/A converter controlled by CPU is used by default for voltage output programming. That source can be calibrated (see the CALibrate subsystem) and provide output within safe limits.
The external voltage programming could be used when fast interaction with an external process is required. For example if tracking output of the connected D.U.T. (i.e. a power supply) is needed the PSU effectively becomes a pre-regulator keeping its output voltage in relation with changes of the D.U.T. output keeping constant difference between connected D.U.T input and output and in that way its max. power dissipation.

Max. D/A converter programmed voltage of 2.5 V would results with MAXimal voltage output regardless of the channel's voltage range (see [Section 8.1](#)). Therefore if external programming source is selected voltage value higher then 2.5 V or lower then zero could produce unexpected results and eventually damage the channel and/or connected load.

The enabled state is EXTernal (1); the disabled state is INTernal (0). Execution of this command also affects bit 13 (RPROG) of the Operation Instrument lsummary register (see [Section 3.3.2](#)).

Self-test operation initiated by *TST? command will reset voltage programming on all PSU channels to the internal/local source.

If external programming source is selected the DIAGnostic[:INFormation]:ADC? Query returns 0.00 value for U_SET.

| Parameters | Name | Type | Range | Default |
|------------------|--|----------|-------------------|----------|
| | <source> | Discrete | INTernal EXTernal | INTernal |
| Return | The query command returns 0 if the local (internal) voltage programming is selected, and 1 if the remote (external) sense is selected. | | | |
| Usage example | VOLT:PROG EXT VOLT:PROG? 1 | | | |
| Errors | 302, "Option not installed" | | | |
| Related Commands | *TST DIAGnostic[:INFormation]:ADC? OUTPut[:STATe] SYSTem:CHANnel[:INFormation]:PROGram? | | | |

5.7.15. [SOURCE[<n>]]:VOLTage:PROTection:DElay[:TIME]

Syntax [SOURCE[<n>]]:VOLTage:PROTection:DElay[:TIME] {<time>|Default}
 [SOURCE[<n>]]:VOLTage:PROTection:DElay[:TIME]? [Default]

Description This command sets the over-voltage protection delay. The over-voltage protection function will not be triggered on the selected output channel during the delay time. After the delay time has expired, the over-voltage protection function will be active. This prevents momentary changes in output status from triggering the over-voltage protection function. Programmed values can range from 0 to 10 seconds. See also [Section 8.1](#)

Return The query command returns the programmed delay time.

| Parameters | Name | Type | Range | Default |
|------------|--------|---------|----------------|---------|
| | <time> | Numeric | 0 – 10 Default | 5ms |

Usage example Get default OVP delay of 50 milliseconds:
 VOLT:PROT:DEL? DEF
 0.050

Errors

Related Commands OUTPut:PROTection:CLEar

5.7.16. [SOURCE[<n>]]:VOLTage:PROTection:STATe

Syntax [SOURCE[<n>]]:VOLTage:PROTection:STATe {<bool>}
 [SOURCE[<n>]]:VOLTage:PROTection:STATe?

Description This command enables or disables the over-voltage protection (OVP) function. The enabled state is ON (1); the disabled state is OFF (0). Since the PSU do not have a dedicated over-voltage protection circuit that can be programmed independently of output current level, entering the CV (constant voltage) mode of operation is used as a trigger to start OVP sequence. When delay time specified with the [SOURCE[<n>]]:VOLTage:PROTection:DElay[:TIME] command expired the output turns off and the Questionable Condition status register OCP bit 8 is set. An error tone will also follow if beeper is enabled (see SYSTem:BEEPer:STATe). [SOURCE[<n>]]:VOLTage:PROTection:TRIPped? command can be used to query whether over-voltage protection occurred on the selected channel.

Return The query command returns 0 if the voltage protection state is OFF, and 1 if the voltage protection state is ON.

| Parameters | Name | Type | Range | Default |
|------------|--------|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |

Usage example VOLT:PROT:STAT?
 0

Errors

Related Commands OUTPut:PROTection:CLEar
 Commands [SOURCE[<n>]]:VOLTage:PROTection:DElay[:TIME]
 [SOURCE[<n>]]:VOLTage:PROTection:TRIPped
 SYSTem:BEEPer:STATe

5.7.17. [SOURCE[<n>]]:VOLTage:PROTection:TRIPped?

Syntax [SOURCE[<n>]]:VOLTage:PROTection:TRIPped?

Description Query whether OVP occurred on the currently selected channel. When protection is tripped bit 8 (OVP) of the Questionable Instrument Isummary register will be set (see [Section 3.4.2](#)).

The [OUTPut:PROTection:CLEar](#) command can be send to clear OVP condition on the selected channel.

Return This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

Usage
example VOLT:PROT:TRIP?
0

Errors

Related
Commands OUTPut:PROTection:CLEar

5.7.18. [SOURce[<n>]]:VOLTage:SENSe[:SOURce]

Syntax [SOURce:]VOLTage:SENSe[:SOURce] {INTernal|EXTernal}
[SOURce:]VOLTage:SENSe[:SOURce]?

Description This command enables or disables remote sensing. The enabled state is EXTernal (1); the disabled state is INTernal (0). Execution of this command also affects bit 12 (RSENSE) of the Operation Instrument Isummary register (see [Section 3.3.2](#)). Self-test operation initiated by [*TST?](#) command will put remote sense on all PSU channels into disable state.

When the BP_OPTION is enabled and the channels are not grouped together (SYSTem:GROup commands scheduled for M3) this command sets LED indicators above binding posts and sense relays in the following manner:

- Turn on/off Sense1+/Sense1- indicators (LED_S1+, LED_S1- or TLC5925 Out5 and Out4) and sense relay K_S1 (TLC5925 Out1) when CH1 is selected
- Turn on/off Sense2+/Sense2- indicators (LED_S2+, LED_S2- or TLC5925 Out12 and Out11) and sense relay K_S2 (TLC5925 Out14) when CH2 is selected

Remote sensing has no effect during CC (Constant Current) operation. Sense+/Sense- indicators (LED_S1+, LED_S1-, LED_S2+, LED_S2-) will not be affected if output state is off (OUTPut OFF command).

| Parameters | Name | Type | Range | Default |
|---------------------|--|----------|-------------------|----------|
| | <source> | Discrete | INTernal EXTernal | INTernal |
| Return | The query command returns 0 if the internal sense is selected, and 1 if the remote (external) sense is selected. | | | |
| Usage example | VOLT:SENS EXT VOLT:SENS? 1 | | | |
| Errors | 302, "Option not installed" | | | |
| Related Commands | *TST OUTPut[:STATe] SYSTem:CHANnel[:INFORMATION]:PROGram? | | | |

5.8. STATus

Status register programming lets you determine the operating condition of the instrument at any time. This subsystem controls the SCPI-defined status-reporting structures. SCPI defines, in addition to those in IEEE 488.2, QUEStionable, OPERation, INSTRument SUMmary and INSTRument registers. These registers conform to the IEEE 488.2 specification and each may be comprised of a condition register, an event register, an enable register. The purpose and definition of the SCPI-defined registers is described in "Volume 1: Syntax and Style". SCPI also defines an IEEE 488.2 queue for status. The queue provides a human readable record of instrument events. The application programmer may individually enable events into the queue.

STATus:PRESet enables errors and disables all other events.

| SCPI command | Description |
|-------------------|--|
| STATus | |
| :OPERation | |
| [:EVENT]? | Returns the value of the Operation Event register |
| :CONDition? | Returns the value of the Operation Instrument Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Event register |
| :INSTRument[<n>] | |
| [:EVENT]? | Returns the value of the Operation Instrument Event register |
| :CONDition? | Returns the value of the Operation Instrument Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Instrument Event register |
| :ISUMmary<n> | |
| [:EVENT]? | Returns the value of the Operation Instrument Isummary Event register |
| :CONDition? | Returns the value of the Operation Instrument Isummary Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Instrument Isummary Event register |
| :PRESet | Presets all enable registers to power-on state |
| :QUEStionable | |
| [:EVENT]? | Returns the value of the Questionable Event register |
| :CONDition? | Returns the value of the Questionable Condition register |
| :ENABLE {<value>} | Enables specific bits in the Questionable Event register |
| :INSTRument[<n>] | |
| [:EVENT]? | Returns the value of the Questionable Instrument Event register |
| :CONDition? | Returns the value of the Questionable Instrument Condition register |
| :ENABLE {<value>} | Enables specific bits in the Questionable Instrument Event register |
| :ISUMmary<n> | |
| [:EVENT]? | Returns the value of the Questionable Instrument Isummary Event register |
| :CONDition? | Returns the value of the Questionable Instrument Isummary Condition register |

:ENABLe {<value>}

Enables specific bits in the Questionable Instrument Summary Event register

5.8.1. STATus:OPERation[:EVENT]?**Syntax** **STATus:OPERation[:EVENT]?****Description** This query returns the value of the read-only Operation Status Event register. The bits are latched and reading the register will clear it. The *CLS command can be also used to clear the register.**Return** The value returned is the binary-weighted sum of all bits set in the register. For example, if bit 9 (decimal value = 512) and bit 13 (decimal value = 8192) are set, this command will return 8704. See table in the [Section 3.3](#) for bits description.**Usage example** If GROUp PARallel (bit 8) is set (next query returns 0 since the first query clears the event register):

STAT:OPER?

256

STAT:OPER?

0

Errors**Related** *CLS**Commands** *STB?

STATus:OPERation:ENABLe

5.8.2. STATus:OPERation:CONDition?**Syntax** **STATus:OPERation:CONDition?****Description** This query returns the value of the read-only Operation Status Condition register.**Return** The value returned is the binary-weighted sum of all bits set in the register. For example, if bit 9 (decimal value = 512) and bit 13 (decimal value = 8192) are set, this command will return 8704. See table in the [Section 3.3](#) for bits description.**Usage example** If GROUp PARallel (bit 8) is set:

STAT:OPER:COND?

256

Errors**Related** STATus:OPERation:ENABLe**Commands****5.8.3. STATus:OPERation:ENABLe****Syntax** **STATus:OPERation:ENABLe {<value>}**
STATus:OPERation:ENABLe?**Description** This command and its query set and read the value of the Operation Status Enable register. The Enable register is a mask for enabling specific bits from the Operation Event register to set the operation summary bit 7 (OPER) of the Status Byte register. This bit is the logical OR of all the Operational Event register bits that are enabled by the Operation Status Enable register.**Return** Query the Operation Status Enable register. The PSU returns a binary-weighted decimal representing the bits set in the enable register.

| Parameters | Name | Type | Range | Default |
|------------|---------|---------|--------------------------------------|----------|
| | <value> | Numeric | A decimal value which corresponds to | PREset=0 |

the binary-weighted sum of the bits in the register (see the table in [Section 3.3](#))

Usage example Enable ISUM (bit 13):
STAT:OPER:ENAB 8192

Errors

Related *CLS
Commands *STB?
STATus:OPERation[:EVENT]?

5.8.4. STATus:OPERation:INSTrument[:EVENT]?

Syntax STATus:OPERation:INSTrument[:EVENT]?

Description This query returns the value of the read-only Instrument Operation Status Event register. The bits are latched and reading the register will clear it. The *CLS command can be also used to clear the register.

Return The value returned is the binary-weighted sum of all bits set in the register. For example, if bit 1 (decimal value = 2) and bit 2 (decimal value = 4) are set, this command will return 6. See table in the [Section 3.3.1](#) for bits description.

Usage example If bit 2 (INST2) is set:
STAT:OPER:INST?
4

Errors

Related *CLS
Commands STATus:PREset

5.8.5. STATus:OPERation:INSTrument:CONDition?

Syntax STATus:OPERation:INSTrument:CONDition?

Description This query returns the value of the read-only Instrument Operation Status Condition register.

Return The value returned is the binary-weighted sum of all bits set in the register. For example, if bit 1 (decimal value = 2) and bit 2 (decimal value = 4) are set, this command will return 6. See table in the [Section 3.3.1](#) for bits description.

Usage example If bit 2 (INST2) is set:
STAT:OPER:INST:COND?
4

Errors

Related STATus:PREset
Commands

5.8.6. STATus:OPERation:INSTrument:ENABLE

Syntax STATus:OPERation:INSTrument:ENABLE {<value>}
STATus:OPERation:INSTrument:ENABLE?

Description Enable bits in the Instrument Operation Status Enable register. The selected bits are then reported to the Operation Status Event register.

Return Query the Instrument Operation Status Enable register. The PSU returns a binary-weighted decimal representing the bits set in the enable register.

| Parameters | Name | Type | Range | Default |
|------------------|--|---------|--|----------|
| | <value> | Numeric | A decimal value which corresponds to the binary-weighted sum of the bits in the register (see the table in Section 3.3.1) | PREset=0 |
| Usage example | Enable INST1 (bit 1) and INST2 (bit 2): STAT:OPER:INST:ENAB 6 | | | |
| Errors | | | | |
| Related Commands | *CLS STATus:PREset | | | |

5.8.7. STATus:OPERation:INSTrument:ISUMmary[<n>][:EVENT]?

| | | | | |
|------------------|---|--|--|--|
| Syntax | STATus:OPERation:INSTrument:ISUMmary[<n>][:EVENT]? | | | |
| Description | This query returns the value of the read-only Instrument Isummary Operation Status Event register for a specific channel of the PSU represented by numeric value [<n>]. When [<n>] is omitted, the system queries the Instrument Isummary Operation Status Event register of the current channel. The bits are latched and reading the register will clear it. The *CLS command can be also used to clear the register. | | | |
| Return | The value returned is the binary-weighted sum of all bits set in the register. See table in the Section 3.3.2 for bits description. | | | |
| Usage example | If bit 8 (CV1) and bit 10 (OE1) on the channel 1 are set (256+1024=1280): STAT:OPER:INST:ISUM1? 1280 | | | |
| Errors | | | | |
| Related Commands | *CLS OUTPut:MODE? | | | |

5.8.8. STATus:OPERation:INSTrument:ISUMmary[<n>]:CONDition?

| | | | | |
|------------------|---|--|--|--|
| Syntax | STATus:OPERation:INSTrument:ISUMmary[<n>]:CONDition? | | | |
| Description | This query returns the value of the read-only Instrument Isummary Operation Status Condition register for a specific channel of the PSU represented by numeric value [<n>]. When [<n>] is omitted, the system queries the Instrument Isummary Operation Status Condition register of the current channel. | | | |
| Return | The value returned is the binary-weighted sum of all bits set in the register. See table in the Section 3.3.2 for bits description. | | | |
| Usage example | If bit 8 (CV1) and bit 10 (OE1) on the channel 1 are set (256+1024=1280): STAT:OPER:INST:ISUM1:COND? 1280 | | | |
| Errors | | | | |
| Related Commands | OUTPut:MODE? | | | |

5.8.9. STATus:OPERation:INSTrument:ISUMmary<n>:ENABLE

| | | | | |
|-------------|---|--|--|--|
| Syntax | STATus:OPERation:INSTrument:ISUMmary[<n>]:ENABLE {<value>} STATus:OPERation:INSTrument:ISUMmary[<n>]:ENABLE? | | | |
| Description | Enable bits in the Instrument Isummary Operation Status Enable register for a specific | | | |

channel of the PSU represented by numeric value [n]. When [n] is omitted, the system queries the Instrument Isummary Operation Status Enable register of the current channel. The selected bits are then reported to the Status Byte.

This command and its query set and read the value of the Operation Status Enable register. The Enable register is a mask for enabling specific bits from the Operation Event register to set the operation summary bit (OPER) of the Status Byte register. This bit (bit 7) is the logical OR of all the Operational Event register bits that are enabled by the Operation Status Enable register

Return Query the Instrument Isummary Operation Status Enable register. The PSU returns a binary-weighted decimal representing the bits set in the enable register.

| Parameters | Name | Type | Range | Default |
|------------|---------|---------|--|----------|
| | <value> | Numeric | A decimal value which corresponds to the binary-weighted sum of the bits in the register (see the table in Section 3.3.2) | PREset=0 |

Usage example The query returns that VOLT1 (bit 0), CURR1 (bit 1) and TEMP1 (bit 4) are set (1+2+16=19):

INST?

CH2

INST CH1

STAT:OPER:INST:ISUM:ENABLE?

19

Errors

Related Commands *CLS
STATus:PREset

5.8.10. STATus:PREset

Syntax [STATus:PREset](#)

Description This command clears all bits in the Enable registers.

Return None

Usage example STAT:PRE

Errors

Related Commands *CLS

5.8.11. STATus:QUEStionable[:EVENT]?

Syntax [STATus:QUEStionable\[:EVENT\]?](#)

Description Query the Questionable Status event register. The bits are latched and reading the register will clear it. The *CLS command can be also used to clear the register.

Return The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register. See table in the [Section 3.4](#) for bits description.

Usage example If the error is detected in RTC (Real-time clock) circuit, the bit 3 (TIME) is set and this command returns 8:

STAT:QUES?

8

Errors

Related *CLS

Commands

5.8.12. STATus:QUEStionable:CONDition?Syntax [STATus:QUEStionable:CONDition?](#)

Description Query the Questionable Status condition register.

Return The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register. See table in the [Section 3.4](#) for bits description.

Usage example If the error is detected in RTC (Real-time clock) circuit, the bit 3 (TIME) is set and this command returns 8:

STAT:QUES:COND?

8

Errors

Related

Commands

5.8.13. STATus:QUEStionable:ENABLESyntax [STATus:QUEStionable:ENABLE {<value>}](#)
[STATus:QUEStionable:ENABLE?](#)Description Enable bits in the Questionable Status Enable register. The selected bits are then reported to the Status Byte.
When <enable value> is set to 0, executing this command will clear the Questionable Status Enable register.

Return Query the Questionable Status Enable register. The PSU returns a binary-weighted decimal representing the bits set in the enable register.

| Parameters | Name | Type | Range | Default |
|------------|---------|---------|--|----------|
| | <value> | Numeric | A decimal value which corresponds to the binary-weighted sum of the bits in the register (see table in Section 3.4) | PREset=0 |

Usage example The query returns that TIME (bit 3), TEMPerature (bit 4) and ISUM (bit 13) are enabled (8+16+8192=8216):

STAT:QUES:ENAB?

8216

Errors

Related *CLS

Commands STATus:PREset

5.8.14. STATus:QUEStionable:INSTrument[:EVENT]?Syntax [STATus:QUEStionable:INSTrument\[:EVENT\]?](#)

Description Query the questionable instrument event register. The bits are latched and reading the register will clear it. The *CLS command can be also used to clear the register.

Return The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register and clears the register. See table in the [Section 3.4.1](#) for bits description.

Usage example Result of the query when INST1 (bit 1) and INST2 (bit 2) are set (2+4=6):
 STAT:QUES:INST?
 6

Errors

Related *CLS
 Commands

5.8.15. STATus:QUESTionable:INSTrument:CONDition?

Syntax STATus:QUESTionable:INSTrument:CONDition?

Description Query the questionable instrument condition register.

Return The PSU returns a decimal value which corresponds to the binary-weighted sum of all bits in the register and clears the register. See table in the [Section 3.4.1](#) for bits description.

Usage example Result of the query when INST1 (bit 1) and INST2 (bit 2) are set (2+4=6):
 STAT:QUES:INST:COND?
 6

Errors

Related
 Commands

5.8.16. STATus:QUESTionable:INSTrument:ENABLE

Syntax STATus:QUESTionable:INSTrument:ENABLE {<value>}
 STATus:QUESTionable:INSTrument:ENABLE?

Description Set the value of the questionable instrument enable register. This register is a mask for enabling specific bits from the questionable instrument event register to set the instrument summary bit 13 (ISUM) of the Questionable Status register. The ISUM bit of the Questionable Status register is the logical OR of all the questionable instrument event register bits that are enabled by the questionable instrument enable register.

Return Query the Questionable Instrument Enable register. The PSU returns a binary-weighted decimal representing the bits set in the enable register.

| Parameters | Name | Type | Range | Default |
|------------|---------|---------|--|----------|
| | <value> | Numeric | A decimal value which corresponds to the binary-weighted sum of the bits in the register (see table in Section 3.4.1) | PREset=0 |

Usage example Set INST1 (bit 1) and INST2 (bit 2):
 STAT:QUES:INST:ENAB 6

Errors

Related *CLS
 Commands

5.8.17. STATus:QUESTionable:INSTrument:ISUMmary[<n>][:EVENT]?

Syntax STATus:QUESTionable:INSTrument:ISUMmary[<n>][:EVENT]?

Description Return the value of the Questionable Instrument Isummary Event register for a specific channel of the PSU represented by numeric value [<n>]. When [<n>] is omitted, the system queries the questionable instrument Isummary enable register of the current chan-

nel. The event register is a read-only register which holds (latches) all events. Reading the Questionable Instrument Isummary Event register clears it. The *CLS command can be also used to clear the register.

When the PSU is operating as a voltage source, bit 1 (CURRENT) is set. When the PSU is operating as a current source, bit 0 (VOLTage) is set. When the output is unregulated (UR), both bits are set (for example, while the output is changing to a new programmed value or when the PSU is sinking instead of sourcing because down-programmer is active with battery with higher voltage then set output is connected).

Return The PSU returns a binary-weighted decimal representing the bits set in the enable register. See table in the [Section 3.4.2](#) for bits description.

Usage example Result of the query when over-current protection (OCP) condition is detected (bit 9):

STAT:QUES:INST:ISUM1?

512

Errors

Related Commands *CLS

5.8.18. STATus:QUEStionable:INSTrument:ISUMmary[<n>]:CONDition?

Syntax STATus:QUEStionable:INSTrument:ISUMmary[<n>]:CONDition?

Description Return the value of the Questionable Instrument Isummary Condition register for a specific channel of the PSU represented by numeric value [<n>]. When [<n>] is omitted, the system queries the questionable instrument Isummary enable register of the current channel.

When the PSU is operating as a voltage source, bit 1 (CURRENT) is set. When the PSU is operating as a current source, bit 0 (VOLTage) is set. When the output is unregulated (UR), both bits are set (for example, while the output is changing to a new programmed value or when the PSU is sinking instead of sourcing because down-programmer is active with battery with higher voltage then set output is connected).

Return The PSU returns a binary-weighted decimal representing the bits set in the enable register. See table in the [Section 3.4.2](#) for bits description.

Usage example Result of the query when over-current protection (OCP) condition is detected (bit 9):

STAT:QUES:INST:ISUM1:COND?

512

Errors

Related Commands

5.8.19. STATus:QUEStionable:INSTrument:ISUMmary[<n>]:ENABLE

Syntax STATus:QUEStionable:INSTrument:ISUMmary[<n>]:ENABLE {<value>}
STATus:QUEStionable:INSTrument:ISUMmary[<n>]:ENABLE?

Description Set the value of the Questionable Instrument Isummary Enable register for a specific channel of the PSU represented by numeric value [<n>]. When [<n>] is omitted, the system queries the Questionable Instrument Isummary Enable register of the current channel. The *CLS command can be used to clear the register.

This register is a mask for enabling specific bits from the Questionable Instrument Isummary Event register to set the Instrument Summary bit (bits 1 and 2) of the Questionable Instrument register. These bits are the logical OR of all the Questionable Instrument Isummary Event register bits that are enabled by the Questionable Instrument Isummary Enable register.

Return Query the value of the Questionable Instrument Isummary Enable register.

| Parameters | Name | Type | Range | Default |
|------------|---------|---------|--|----------|
| | <value> | Numeric | A decimal value which corresponds to the binary-weighted sum of the bits in the register (see table in Section 3.4.2) | PREset=0 |

Usage example Enable bits for all events on channel 2 – VOLT2 (bit 0, value=1), CURR (bit 1, decimal value=2), TEMP2 (bit 4, value 16), OVP1 (bit 8, value=256), OCP2 (bit 9, value=512), OPP2 (bit 10, value=1024), therefore the enable value is 1+2+16+256+512+1024=1811:
STAT:QUES:INST:ISUM2:ENAB 1811

Errors

Related *CLS

Commands STATus:PREset

5.9. SYSTem

System commands control system functions that are not directly related to output control, measurement, or status functions.

| SCPI command | Description |
|--|---|
| SYSTem | |
| :BEEPer[:IMMediate] | Issues a single beep immediately |
| :STATe {<bool>} | Enables beeper function |
| :CAPability? | Returns an <instrument_specifier> |
| :CHANnel | |
| [:COUNT]? | Returns the number of output channels |
| :INFOrmation | |
| :CURRent? | Returns output current capability |
| :POWer | Returns output power capability |
| :PROGram | Returns programmable features of the channel |
| :VOLTage | Returns output voltage capability |
| :MODEl? | Returns the channel model identification |
| CPU | |
| :INFOrmation | |
| :ETHernet | |
| :TYPE? | Returns the type of Ethernet controller |
| :TYPE? | Returns the type of CPU |
| :MODEl? | Returns the control board model identification |
| :OPTion? | Returns information about installed options on the control board |
| :DATE {<yyyy>,<mm>,<dd>} | Sets the date of the system clock |
| :ERRor | |
| [:NEXT]? | Queries and clears errors from the error queue |
| :COUNT? | Queries the error/event queue for the number of unread items |
| :POWer {<bool>} | Enters the PSU into the Stand-by mode |
| :TEMPerature | |
| :PROTection | |
| [:HIGH] | |
| [:LEVel] {<temperature>[, <sensor>]} | Sets the OTP value |
| :CLEar[, {<sensor>}] | Clears the latched protection status of the over-temperature protection (OTP) |
| :DELay | Sets time-out period |
| [:TIME] {<delay>[, <sensor>]} | Sets the OTP programming delay |
| :STATe {<bool>[, <sensor>]} | Enables/disables OTP on the selected temperature sensor |
| :TRIPped? [<sensor>] | Returns status of OTP activation |
| :TIME {<hh>,<mm>,<ss>} | Sets the time of the system clock |
| :VERSion? | Returns the SCPI version number |

5.9.1. SYSTem:BEEPer

Syntax `SYSTem:BEEPer[:IMMediate]`

Description This command issues a single beep immediately.

Usage `SYST:BEEP`

example

5.9.2. SYSTem:BEEPer:STATe

Syntax `SYSTem:BEEPer:STATe {<bool>}`
`SYSTem:BEEPer:STATe?`

Description When the beeper is enabled, the PSU generates audible sound in any of the following situations:

- the power is turns on or off (see `SYSTem:POWer`),
- when error occurs during front panel operation or remote operation (see [Section 7](#) for the list of error messages),
- self-test is failed and
- any of the protection function is “tripped”

| Parameters | Name | Type | Range | Default |
|------------------|---|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | ON |
| Usage example | <code>SYST:BEEP:STAT ON</code> <code>SYST:BEEP:STAT?</code> 1 | | | |
| Related Commands | <code>SYSTem:BEEPer[:IMMediate]</code> <code>SYSTem:POWer</code> | | | |

5.9.3. SYSTem:CAPability?

Syntax `SYSTem:CAPability?`

Description This query returns the PSU's capabilities and outputs the appropriate specifiers. See also SCPI Volume 4: Section 7.1, 1.4.1, 7.2.1, 7.2.2, and 7.2.3

Usage example `SYSTem:CAPability?`
`DCSUPPLY WITH (MEASURE|MULTIPLE|TRIGGER)`

5.9.4. SYSTem:CHANnel[:COUNT]?

Syntax `SYSTem:CHANnel[:COUNT]?`

Description This query returns the number of output channels in a mainframe.

Usage example `SYSTem:CHANnel?`
2

Related Commands `INSTRument[:SElect]`
`INSTRument:NSElect`

5.9.5. SYSTem:CHANnel:INFOrmation:CURRent?

Syntax `SYSTem:CHANnel:INFOrmation:CURRent? [<channel>]`

Description Use this query to get currently selected channel output current capability.

| Parameters | Name | Type | Range | Default |
|------------|-----------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |

Usage SYST:CHAN:INFO:CURR?
 example 5.00

5.9.6. SYSTem:CHANnel:INFOrmation:POWEr?

Syntax SYSTem:CHANnel:INFOrmation:POWEr? [<channel>]

Description Use this query to get currently selected channel output power capability.

| Parameters | Name | Type | Range | Default |
|---------------|-------------------------------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |
| Usage example | SYST:CHAN:INFO:POW? 160.00 | | | |

5.9.7. SYSTem:CHANnel:INFOrmation:PROGram?

Syntax SYSTem:CHANnel:INFOrmation:PROGram? [<channel>]

Description This query returns names of all channel board functionality that can be controlled by firmware. Depending of the board model (see the SYSTem:CHANnel:MODEl? query) various combination of the following features can be returned:

- Volt – program the output voltage while channel is in the CV mode of operation (see [SOURce[<n>]]:VOLTage and APPLy commands)
- Current – program the output current while channel is in the CC mode of operation (see [SOURce[<n>]]:CURRent and APPLy commands)
- Power – set max. allowed output power regardless of the channel mode of operation (see [SOURce[<n>]]:POWEr:LIMit)
- OE – set channel power output (see OUTPut[:STATe])
- Dprog – control down-programmer circuit (see OUTPut:DPRog)
- LRipple – set low power mode of operation when SMPS pre-regulator is switched off (see OUTPut:LRIPple)
- Rprog – control output voltage programming source (see [SOURce[<n>]]:VOLTage:PROGram[:SOURce])

| Parameters | Name | Type | Range | Default |
|------------------|---|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |
| Usage example | SYST:CHAN:INFO:PROG? CH1 "Volt", "Current", "Power", "OE", "DProg", "LRipple", "Rprog" | | | |
| Related Commands | APPLy OUTPut:DPRog OUTPut:MODE? OUTPut[:STATe] OUTPut:LRIPple [SOURce[<n>]]:CURRent [SOURce[<n>]]:POWEr:LIMit [SOURce[<n>]]:VOLTage [SOURce[<n>]]:VOLTage:PROGram[:SOURce] SYSTem:CHANnel:MODEl? | | | |

5.9.8. SYSTem:CHANnel:INFOrmation:VOLTage?

Syntax SYSTem:CHANnel:INFOrmation:VOLTage? [<channel>]

Description Use this query to get currently selected channel output voltage capability.

| Parameters | Name | Type | Range | Default |
|------------|-----------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | – |

Usage SYST:CHAN:INFO:VOLT?
 example 40.00

5.9.9. SYSTem:CHANnel:MODeI?

Syntax SYSTem:CHANnel:MODeI? [<channel>]

Description This query returns the model identification string of the specified channel.

| Parameters | Name | Type | Range | Default |
|------------|---------------------------------|----------|---------|---------|
| | <channel> | Discrete | CH1 CH2 | — |
| Usage | SYST:CHAN:MOD? | | | |
| example | "Power_r5B6b", "Postreg_r4B43a" | | | |
| | SYST:CHAN:MOD? CH2 | | | |
| | "Postreg_r4B43a" | | | |

5.9.10. SYSTem:CPU:INFOrmation:ETHernet:TYPE?

Syntax SYSTem:CPU:INFOrmation:ETHernet:TYPE?

Description This query returns the name of Ethernet controller installed on the control board (Arduino shield). If simulator is used it returns "Simulator".

Return The information will be returned as a list of quoted strings.

Usage SYST:CPU:INFO:ETH:TYPE?
 example "W5500"

5.9.11. SYSTem:CPU:INFOrmation:TYPE?

Syntax SYSTem:CPU:INFOrmation:TYPE?

Description This query returns the name of CPU installed on the control board (Arduino shield). If simulator is used it returns "Simulator".

Usage SYST:CPU:TYPE?
 example "Due"

Related *IDN?
 Commands

5.9.12. SYSTem:CPU:MODeI?

Syntax SYSTem:CPU:MODeI?

Description This query returns the name of the control board (Arduino shield). If simulator is used it returns "Simulator" and its version.

Usage If revision 1 Arduino shield is detected:
 example SYST:CPU:MODeI?

"Arduino", "r1B9"

If firmware is running on simulator:

SYST:CPU:MODeI?

"Simulator", "M2.0"

Related SYSTem:CPU:OPTion?
 Commands

5.9.13. SYSTem:CPU:OPTion?

| | |
|------------------|--|
| Syntax | SYSTem:CPU:OPTion? |
| Description | This query returns a list of all installed options on the control board. |
| Return | The information will be returned as a list of quoted strings. |
| Usage example | SYST:CPU:MODEl? "BPost", "EEPROM", "RTC", "SDcard", "Ethernet", "USB", "Ext_trig", "Ext_prog", "Watchdog", "Fan" |
| Related Commands | SYSTem:CPU:MODEl? |

5.9.14. SYSTem:DATE

| | | | | |
|------------------|--|---------|-------------|---------|
| Syntax | SYSTem:DATE {<yyyy>,<mm>,<dd>} SYSTem:DATE? | | | |
| Description | <p>Sets the date of the system clock (RTC). Specify the year, month, and day. The self-test procedure compare date and time stored in RTC registers with values stored in the non-volatile memory (EEPROM). When the later is greater then former or any of them lost integrity (i.e. any of value is outside allowed range: for example seconds are higher then 60 or months are higher then 12, etc.) self-test will failed. The *TST? will return 1 and detailed report could be queried using the DIAGnostic:TEST? command.</p> <p>The bit 3 (TIME) of the Questionable Status register will be set (see Section 3.4) if date-time self-test failed or datetime was never set.</p> | | | |
| Return | Query the current date of the system clock in YYYY, MM, DD format. | | | |
| Parameters | Name | Type | Range | Default |
| | <yyyy> | Numeric | 2000 – 2099 | – |
| | <mm> | Numeric | 1 – 12 | – |
| | <dd> | Numeric | 1 – 31 | – |
| Usage example | SYST:DATE? 2015, 10, 24 | | | |
| Errors | | | | |
| Related Commands | *TST? DIAGnostic[:INfOrmation]:TEST? SYSTem:TIME | | | |

5.9.15. SYSTem:ERRor

| | |
|---------------|--|
| Syntax | SYSTem:ERRor[:NEXT]? |
| Description | This query command reads and clear errors from the error queue. A record of up to 20 errors can be stored in the PSU's error queue. See also "Error Messages" in Section 7 . Errors are retrieved in first-in-first-out (FIFO) order. The first error returned is the first error that was stored. The PSU beeps once each time an error is generated. The error queue is cleared when power has been off or after a *CLS command. |
| Return | SYSTem:ERRor[:NEXT]? queries and clears the error messages in the error queue. The query returns the number and content of the error message. |
| Usage example | SYST:ERR? -113,"Undefined header" |

| | |
|------------------|--|
| Errors | <p>If more than 20 errors have occurred, the last error stored in the queue (the most recent error) is replaced with:</p> <p>-350, "Queue overflow"</p> <p>No additional errors are stored until you remove errors from the queue.</p> |
| Related Commands | <p>*CLS</p> <p>*RST</p> <p>SYSTem:ERRor:COUNT</p> |

5.9.16. SYSTem:ERRor:COUNT?

| | |
|------------------|--|
| Syntax | SYSTem:ERRor:COUNT? |
| Description | This query command queries the error/event queue for the number of unread items. As errors and events may occur at any time, more items may be present in the queue at the time it is actually read. |
| Return | |
| Usage example | <p>SYST:ERR:COUN?</p> <p>10</p> |
| Errors | |
| Related Commands | <p>*CLS</p> <p>*RST</p> <p>SYSTem:ERRor[:NEXT]</p> |

5.9.17. SYSTem:POWer

| | | | | |
|------------------|--|----------|------------|---------|
| Syntax | <p>SYSTem:POWer {<bool>}</p> <p>SYSTem:POWer?</p> | | | |
| Description | <p>This command controls powering down and powering up sequence of the main transformer that supply all PSU channels. The "Stand-by" indicator (LED_PWR, TLC5925 Out15) will be switched on when the PSU enters the Stand-by mode.</p> <p>*RST command will perform SYSTem:POWer ON that follows SYSTem:POWer OFF after 5 seconds.</p> <p><i>Arduino Shield without BP option (parameter BP_OPTION is false) has LED_PWR connected to the digital output LED_PWR (pin 66).</i></p> | | | |
| Return | Query returns PSU power Stand-by status. | | | |
| Parameters | Name | Type | Range | Default |
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| Usage example | <p>SYST:POW:STAN ON</p> <p>SYST:POW:STAN?</p> <p>1</p> | | | |
| Errors | | | | |
| Related Commands | <p>*RST</p> <p>*TST?</p> <p>SYSTem:BEEP:STATe</p> | | | |

5.9.18. SYSTem:TEMPerature:PROTection[:HIGH][:LEVel]

| | |
|-------------|---|
| Syntax | <p>SYSTem:TEMPerature:PROTection[:HIGH][:LEVel] {<temperature>[, <sensor>]}</p> <p>SYSTem:TEMPerature:PROTection[:HIGH][:LEVel]? [<sensor>]</p> |
| Description | Set the over-temperature protection (OTP) value in degrees Celsius (°C) of the selected temperature sensor. When the over-temperature protection function of the specified tem- |

perature sensor is enabled (SYSTem:TEMPerature:PROTection[:HIGH]:STATe), one of the following action will be performed when the temperature exceeds the over-temperature protection value currently set:

- MAIN – Switch off power of the main transformer and set bit 4 of the Questionable Status register

If MAIN temperature sensor cause over-temperature condition an error tone will also follow if beeper is enabled (see SYSTem:BEEPer:STATe).

SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped? command can be used to query whether over-temperature protection occurred on the selected temperature sensor.

Return Query the over-temperature protection (OTP) value of the selected temperature sensor.

| Parameters | Name | Type | Range | Default |
|---------------|--|---------------|---------|---------|
| | <temperature> | Numeric (NR2) | 0 – 100 | 70 |
| | <sensor> | Discrete | [MAIN] | MAIN |
| Usage example | TEMP:PROT:50.5, MAIN TEMP:PROT? MAIN 50.50 | | | |

Errors

Related *RST

Commands SYSTem:TEMPerature:PROTection[:HIGH]:STATe
SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped?

5.9.19. SYSTem:TEMPerature:PROTection[:HIGH]:CLEAr

Syntax SYSTem:TEMPerature:PROTection[:HIGH]:CLEAr[, {<sensor>}]

Description This command clears the latched protection status when an over-temperature is detected.
All conditions that generate the fault must be removed before the latched status can be cleared. The output is restored to the state it was in before the fault condition occurred.

Return None

| | Name | Type | Range | Default |
|---------------|---------------|----------|--------|---------|
| | <sensor> | Discrete | [MAIN] | MAIN |
| Usage example | TEMP:PROT:CLE | | | |

Errors

Related SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped
Commands

5.9.20. SYSTem:TEMPerature:PROTection[:HIGH]:DELay[:TIME]

Syntax SYSTem:TEMPerature:PROTection[:HIGH]:DELay[:TIME] {<delay>[, <sensor>]}
SYSTem:TEMPerature:PROTection[:HIGH]:DELay[:TIME]?[, {<sensor>}]

Description This command sets the over-temperature protection delay. The over-temperature protection function will not be triggered during the delay time. After the delay time has expired, the over-temperature protection function will be active.
Programmed values can range from 0 to 300 seconds. See also [Section 8.1](#)

Return The query returns programmed over-temperature protection delay.

| Parameters | Name | Type | Range | Default |
|------------|----------|----------|-----------------|---------|
| | <delay> | Numeric | 0 – 300 seconds | 10 |
| | <sensor> | Discrete | [MAIN] | MAIN |

Usage TEMP:PROT:DEL 30, S2

example

Errors

Related *RST

Commands SYSTem:TEMPerature:PROTection[:HIGH][:LEVel]

5.9.21. SYSTem:TEMPerature:PROTection[:HIGH]:STATe

Syntax SYSTem:TEMPerature:PROTection[:HIGH]:STATe {<bool>[, <sensor>]}
 SYSTem:TEMPerature:PROTection[:HIGH]:STATe?[, {<sensor>}]

Description This command enables or disables the over-temperature protection (OTP) function. The enabled state is ON (1); the disabled state is OFF (0). If the over-temperature protection function is enabled and the measured output power reach value set by [SOURce[<n>]]:POWer:PROTection[:LEVel] the output is disabled and the Questionable Condition status register OPP bit 10 is set.

Return The query command returns 0 if the current protection state is OFF, and 1 if the current protection state is ON.

| Parameters | Name | Type | Range | Default |
|------------|----------|----------|------------|---------|
| | <bool> | Discrete | ON OFF 0 1 | OFF |
| | <sensor> | Discrete | [MAIN] | MAIN |

Usage example TEMP:PROT:STAT? S1
 0

Errors

Related *RST

Commands SYSTem:TEMPerature:PROTection[:HIGH]:CLEar

5.9.22. SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped?

Syntax SYSTem:TEMPerature:PROTection[:HIGH]:TRIPped?[, {<sensor>}]

Description Query whether OTP occurred on the selected temperature sensor. When protection is tripped bit 4 (TEMPerature) of the Questionable Status register will be set (see [Section 3.4](#)).
 The SYSTem:TEMPerature:PROTection[:HIGH]:CLEar command can be send to clear OTP condition caused by the selected temperature sensor.

Return This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

| Parameters | Name | Type | Range | Default |
|------------|----------|----------|--------|---------|
| | <sensor> | Discrete | [MAIN] | MAIN |

Usage example TEMP:PROT:TRIP?
 0

Errors

Related SYSTem:TEMPerature:PROTection[:HIGH]:CLEar

Commands SYSTem:TEMPerature:PROTection[:HIGH]:STATe

5.9.23. SYSTem:TIME

Syntax SYSTem:TIME {<hh>,<mm>,<ss>}
 SYSTem:TIME?

Description Sets the time of the system clock (RTC). Specify the hours, minutes, and seconds. The self-test procedure compare date and time stored in RTC registers with values stored in the non-volatile memory (EEPROM). When the later is greater then former or any of them lost integrity (i.e. any of value is outside allowed range: for example seconds are higher then 60 or months are higher then 12, etc.) self-test will failed. The *TST? will

return 1 and detailed report could be queried using the DIAGnostic:TEST? command.

The bit 3 (TIME) of the Questionable Status register will be set (see [Section 3.4](#)) if date-time self-test failed or datetime was never set.

Return Query the current time of the system clock in HH, MM, SS format.

| Parameters | Name | Type | Range | Default |
|------------------|--|---------|--------|---------|
| | <hh> | Numeric | 0 – 23 | – |
| | <mm> | Numeric | 0 – 59 | – |
| | <ss> | Numeric | 0 – 59 | – |
| Usage example | SYST:TIME? 15, 10, 33 | | | |
| Errors | | | | |
| Related Commands | *TST? DIAGnostic[:INFormation]:TEST? SYSTem:DATE | | | |

5.9.24. SYSTem:VERSion?

Syntax **SYSTem:VERSion?**

Description This command returns the version of the SCPI (Standard Commands for Programmable Instruments) standard with which the instrument is in compliance

Return The command returns a string in the form “YYYY.V”, where YYYY represents the year of the version and V represents a version for that year.

Usage example
SYST:VERS?
1999.0

6. Device-specific (unclassified) commands

The commands in this section are device-specific to the PSU. They are not included in the 1999.0 version of the SCPI standard. However, these commands are designed with the SCPI standard in mind, and they follow all of the command syntax rules defined by the standard.

6.1. APPLY

The APPLY command provides the most straightforward method to program the PSU over the remote interface.

Syntax `APPLy {<channel>}
[,<voltage>|MINimum|MAXimum|DEF[,<current>|MINimum|MAXimum|DEFault]]
APPLy? {<channel>}[, <param>]`

Description This command is a combination of [INSTrument:SElect](#) (or [INSTrument:NSElect](#)), [\[SOURce\[<n>\]:VOLTage](#) and [\[SOURce\[<n>\]:CURRent](#) commands.

The APPLY changes the PSU's output to the newly programmed values only if the programmed values are valid within the presently selected range. An execution error will occur if the programmed values are not valid within the selected range. You can substitute MINimum, MAXimum, or DEFault in place of a specific value for the voltage and current parameters (see table below).

| Parameters | Name | Type | Range | Default |
|------------|-------------|----------|--|---------|
| | {<channel>} | Discrete | CH1 CH2 | – |
| | <voltage> | Numeric | 0 to maximum, MIN DEF MAX UP DOWN The maximum value is dependent on the PSU voltage rating. See Section 8.1 | – |
| | <current> | Numeric | 0 to MAXimum, MIN DEF MAX UP DOWN The maximum value is dependent on the PSU current rating. See Section 8.1 | – |
| | <param> | Discrete | CURR VOLT | – |

Return `APPLy?` query the voltage/current of the specified channel.

Usage example Set the voltage and current of CH1 to 35.5V and 0.5A respectively:

```
APPL CH1, 35.5, 0.5
```

Query the voltage and current settings of the first channel:

```
APPL? CH1
```

```
CH1:50V/3A, 35.500, 0.500
```

Query only current setting of the second channel:

```
APPL? CH2, CURR
```

```
0.25
```

Errors -221, "Power limit exceeded"
-222, "Data out of range"

Related Commands `INSTrument:NSElect`
`INSTrument[:SElect]`
`[SOURce[<n>]:VOLTage[:LEVel][:IMMediate][:AMPLitude]`
`[SOURce[<n>]:CURRent[:LEVel][:IMMediate][:AMPLitude]`

7. Error messages

The system-defined error/event numbers are chosen on an enumerated ("1 of N") basis. The SCPI-defined error/event numbers and the <error/event_description> portions of the full queue item are listed here. The first error/event described in each class (for example, -100, -200, -300, -400) is a "generic" error.

7.1. Command Error [-199, -100]

An <error/event number> in the range [-199, -100] indicates that a syntax error has been detected by the PSU's parser. The occurrence of any error in this class causes the command error bit (CME, bit 5) in the Standard Event Status Register (see [Section 3.1](#)) to be set.

| Return string | Description |
|------------------------------------|--|
| 0, "No error" | The queue is completely empty. Every error/event in the queue has been read or the queue was purposely cleared by power-on, *CLS, etc. |
| -100, "Command error" | Generic syntax error. |
| -101, "Invalid character" | An invalid character was found in the command string. You may have inserted a character such as #, \$, or % in the command keyword or within a parameter. Example: OUTP:STAT #ON |
| -103, "Invalid separator" | An invalid separator was found in the command string. You may have used a comma instead of a colon, semicolon, or blank space, or you may have used a blank space instead of a comma. Example: TRIG:SOUR, BUS |
| -104, "Data type error" | The wrong parameter type was found in the command string. You may have specified a number where a string was expected, or vice versa. Example (password is not a quoted string): CAL ON, 123 |
| -108, "Parameter not allowed" | More parameters were received than expected for the command. You may have entered an extra parameter, or added a parameter to a command that does not accept a parameter. Example: INST CH1, CH2 |
| -109, "Missing parameter" | Fewer parameters were received than expected for the command. You omitted one or more parameters that are required for this command. Example: APPL |
| -113, "Undefined header" | A command was received that is not valid for this PSU. You may have misspelled the command or it may not be a valid command. If you are using the short form of the command, remember that it may contain up to four letters. Example: MEASU:CURR? |
| -114, "Header suffix out of range" | The numeric suffix attached to a command header is not one of the allowable values. |

| | |
|-----------------------------|---|
| | Example: STAT:QUES:INST:ISUM3? |
| -131, "Invalid suffix" | A suffix was incorrectly specified for a numeric parameter. You may have misspelled the suffix. Example (use A instead of V): VOLT 3A |
| -138, "Suffix not allowed" | A suffix was received following a numeric parameter which does not accept a suffix. Example (SEC is not a valid suffix): STAT:QUES:ENAB 10 SEC |
| -151, "Invalid string data" | An invalid character string was received. Check to see if you have enclosed the character string in single or double quotes. Example: DISP:TEXT 'ON |

7.2. Execution Error [-299, -200]

An <error/event number> in the range [-299, -200] indicates that an error has been detected by the PSU's execution control block. The occurrence of any error in this class cause the execution error bit (EXE, bit 4) in the Standard Event Status Register (see [Section 3.1](#)) to be set. One of the following events has occurred:

- A <PROGRAM DATA> element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the PSU's capabilities.
- A valid program message could not be properly executed due to some PSU condition.

Execution errors will be reported by the PSU after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, will not be reported as an execution error. Events that generate execution errors will not generate Command Errors, device-specific errors, or Query Errors; see the other error definitions in this section.

| Return string | Description |
|---------------------------------|--|
| -200, "Execution error" | This is the generic execution error when more specific error is not assigned in the case that command execution failed. |
| -222, "Data out of range" | A numeric parameter value is outside the valid range for the command. Example: VOLT 166 |
| -223, "Too much data" | A character string was received but could not be executed because the string length was more than 32 characters. This error can be generated by the CALibration:REMark command. |
| -224, "Illegal parameter value" | A discrete parameter was received which was not a valid choice for the command. You may have used an invalid parameter choice. Example: VOLT ON |
| -240, "Hardware error" | Command or query could not be executed because failure is detected during power-up self-test. Use *TST? command to query self-test results. See also Section 7.3.1 |
| -241, "Option not installed" | Command or query could not be executed because of missing PSU hardware. Example (remote sense cannot be activated when BP_OP- |

| | |
|--------------------------------|---|
| | TION is false): OUTP:SENS ON |
| -242, "Channel fault detected" | POWERGOOD signal failure is detected on any of the installed channel. If such condition is happened the PSU will be immediately put into the stand-by mode. |

7.3. Device-Specific Error [-399, -300], [1, 32767]

An <error/event number> in the range [-399, -300] or [1, 32767] indicates that the PSU has detected an error which is not a command error, a query error, or an execution error; some PSU operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. The occurrence of any error in this class cause the device-specific error bit (DDE, bit 3) in the Standard Event Status Register (see [Section 3.1](#)) to be set.

Events that generate device-specific errors do not generate command errors, execution errors, or query errors; see the other error definitions in this section.

| Return string | Description |
|-------------------------------------|---|
| -310, "System error" | Internal firmware error. |
| -330, "Self-test failed" | The PSU's complete self-test failed from the remote interface (*TST? command). In addition to this error, more specific self-test errors are also reported. See also Section 7.3.1 |
| -350, "Queue overflow" | The error queue is full because more than 16 errors have occurred. No additional errors are stored until you remove errors from the queue. The error queue is cleared when power has been turned off, or after a *CLS (clear status) command has been executed. |
| -363, "Input buffer overrun" | Input buffer overrun. Serial or Ethernet port input buffer overflows with data. |
| 100, "Channel not found" | Non existing channel number is specified. Example: SOUR3:VOLT? |
| 101, "Calibration mode is off" | Calibration is not enabled. The PSU will not accept calibration commands. |
| 102, "Invalid cal password" | The calibration password is incorrect. |
| 104, "Bad sequence of cal commands" | Calibration commands have not been entered in the proper sequence. |
| 105, "Cal password too long" | A calibration password was received which contained more than 16 characters. |
| 106, "Cal password too short" | A calibration password was received which contained less than 4 characters. |
| 107, "Cal value out of range" | The specified calibration value (CALibration:CURRent:DATA or CALibration:VOLTage:DATA) is not valid for the present measurement function and range. |
| 108, "Cal output disabled" | Calibration is aborted by sending the OUTP OFF command during an output calibration. |
| 109, "Invalid cal data" | One or more calibration data have value that prohibit calibration parameters calculation. For example MID value is lower than MIN, or MIN is higher than MAX, or MID value is out of accepted tolerance that it cannot be predicted with newly calculated calibration parameters. |

| | |
|--|---|
| 110, "Cal params missing or corrupted" | Calibration parameters activation initiated by CALibration:STATe ON, "<password>" failed because calibration has never been conducted or existing parameters are corrupted. |
| 111, "No new cal data exists" | Attempt to save calibration data with the CAL:SAVE command while no new calibration data are entered. |
| 150, "Power limit exceeded" | Product of voltage and current exceeds channel power limitation. For example if channel power limit is 160 W and the following sequence is executed: VOLT 38 CURR 4.4 |
| 201, "Cannot execute before clearing protection" | Command such as OUTP ON cannot be executed on the channel where one or more protections are tripped. |
| 270, "CH1 ADC timeout detected" | Channel is switched off after 3 consecutive ADC timeouts are detected. |

7.3.1. Self-Test Error Messages

During power-up the PSU will start self-test sequence when communication with all SPI devices that is marked as installed will be established. Scope of self-test depends of device capability and it could vary from simple reading device registers and waiting for expecting response to more complex operation such as set DAC registers and read back set values using ADC (if ADC test passed). Every test failure will be announced by error beep, and one error message per failed test will be inserted into error queue. That also cause the device-specific error bit (DDE, bit 3) in the Standard Event Status Register (see [Section 3.1](#)) to be set.

| Return string | Description |
|------------------------------------|---|
| 210, "CH1 IOEXP test failed" | Communication with Channel 1 I/O expander is not possible. I/O expander is not functional or SPI 10-pin ribbon cable is not connected. |
| 211, "CH2 IOEXP test failed" | Communication with Channel 2 I/O expander is not possible. I/O expander is not functional or SPI 10-pin ribbon cable is not connected. |
| 220, "CH1 ADC test failed" | Communication with Channel 1 ADC is not possible. Values written into various registers are not equal to the returned values or SPI 10-pin ribbon cable is not connected. |
| 221, "CH2 ADC test failed" | Communication with Channel 2 ADC is not possible. Values written into various registers are not equal to the returned values or SPI 10-pin ribbon cable is not connected. |
| 230, "CH1 DAC test failed" | Channel 1 DAC is not functional, communication is failed or difference between set and acquired DAC output value is too high when CPU sends a test voltage data to the DAC and converts the DAC output to digital data (using ADC I_SET input). |
| 231, "CH2 DAC test failed" | Channel 2 DAC is not functional, communication is failed or difference between set and acquired DAC output value is too high when CPU sends a test voltage data to the DAC and converts the DAC output to digital data (using ADC I_SET input). |
| 240, "External EEPROM test failed" | Non-volatile memory on the Arduino board checksum test failed. |
| 250, "RTC test failed" | RTC on the Arduino board is not present or date or time values are not valid or last datetime stored into non-volatile memory is greater then RTC datetime. |
| 260, "Ethernet test failed" | Ethernet controller on the Arduino board test failed. |

```
630, "Fan test failed"
```

For example if SPI-bus cable is incidentally not connected or wrongly wired on channel 1 two errors will be generated and placed into error queue what can be check using the following command sequence:

```
SYST:ERR:COUN?
```

```
2
```

```
SYST:ERR?
```

```
210, "CH1 IOEXP test failed"
```

```
SYST:ERR?
```

```
220, "CH1 ADC test failed"
```

8. Parameters and settings

8.1. Programming parameters

The PSU firmware could be used to control channels with various characteristics. For example voltage range could be from 0 to 30 V, 0 to 40 V or 0 to 50 V and current range could vary from 0 to 3.12 A, 0 to 4.16 A or 0 to 5 A. It's also possible to mix two channels with different voltage and current ranges e.g. 0 – 40 V/0 – 5 A and 0 – 50 V/0 – 3.12 A. Use [*IDN?](#) command to find out what channels are defined in the firmware.

8.1.1. Voltage

| Programming range / Model | 0 – 30 V / 30 | 0 – 40 V / 40 | 0 – 50 V / 50 |
|------------------------------|---------------|---------------|---------------|
| MAXimum [V] | 30 | 40 | 50 |
| MINimum [V] | 0 | 0 | 0 |
| DEFault [V] | 0 | 0 | 0 |
| Value after *RST or *TST? | 0 | 0 | 0 |
| STEP MINimum [V] | 10m | 10m | 10m |
| STEP MAXimum [V] | 5 | 5 | 5 |
| STEP DEFault [V] | 100m | 100m | 100m |
| PROtection DELay MINimum [s] | 0 | 0 | 0 |
| PROtection DELay MAXimum [s] | 10 | 10 | 10 |
| PROtection DELay DEFault [s] | 5m | 5m | 5m |
| CALibration VALue MINimum[V] | 0.2 | 0.2 | 0.2 |
| CALibration VALue MIDdle[V] | 14.1 | 19.1 | 24.1 |
| CALibration VALue MAXimum[V] | 28 | 38 | 48 |

8.1.2. Current

| Programming range / Model | 0 – 3.12 A / 03 | 0 – 5 A / 05 |
|------------------------------|-----------------|--------------|
| MAXimum [A] | 3.12 | 5 |
| MINimum [A] | 0 | 0 |
| DEFault [A] | 0 | 0 |
| Value after *RST or *TST? | 0 | 0 |
| STEP and STEP MINimum [A] | 10m | 10m |
| STEP MAXimum [A] | 1 | 1 |
| STEP DEFault [A] | 50m | 50m |
| PROtection DELay MINimum [s] | 0 | 0 |
| PROtection DELay MAXimum [s] | 10 | 10 |
| PROtection DELay DEFault [s] | 20m | 20m |
| CALibration VALue MINimum[A] | 50m | 50m |
| CALibration VALue MIDdle[A] | 1.525 | 2.425 |
| CALibration VALue MAXimum[A] | 3 | 4.8 |

8.1.3. Power

The total number of possible combinations for power parameters are 6 of which two typical examples are stated below:

| Programming range / Model example | 0 – 50 V, 0 – 3.12 A / 05/03 | 0 – 40 V, 0 – 5 A / 40/05 |
|-----------------------------------|------------------------------|---------------------------|
| MINimum [W] | 5 | 10 |
| DEFault [W] | 100 | 150 |
| MAXimum [W] | 150 | 160 |
| PROtection LEVel DEFault [W] | 100 | 150 |
| PROtection DELay MINimum [s] | 1 | 1 |
| PROtection DELay MAXimum [s] | 300 | 300 |
| PROtection DELay DEFault [s] | 10 | 10 |

8.2. Reset Settings (*RST)

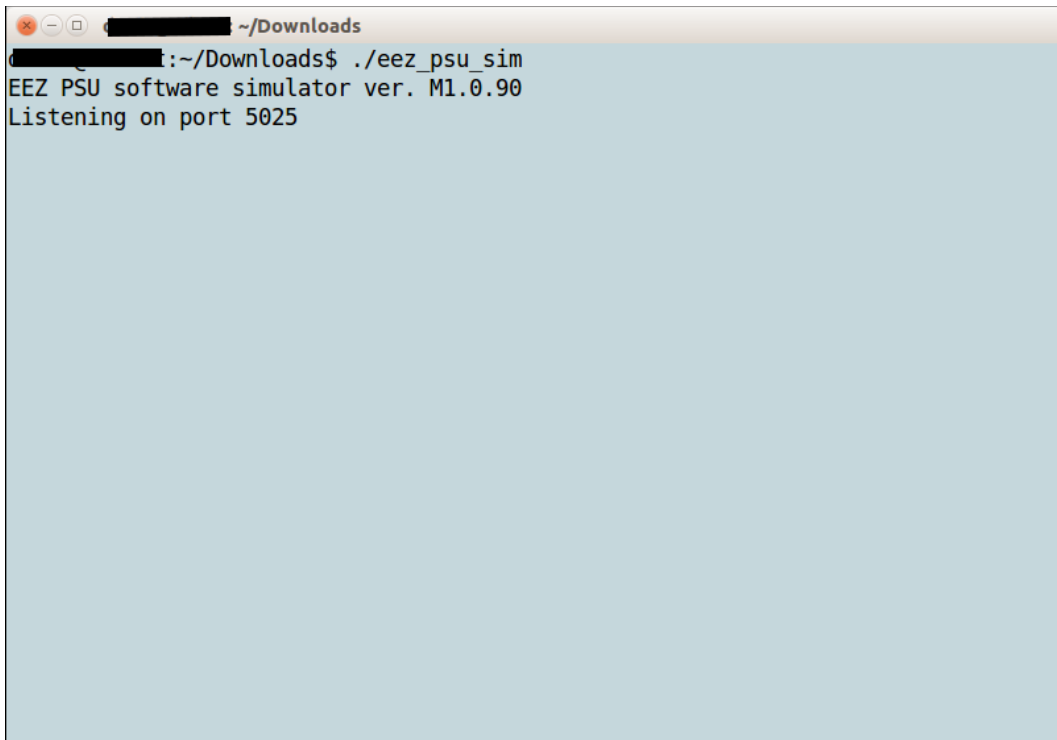
At power-on or after execution of the [*RST](#) common command, device settings will be set to states that are listed in the table that follows. See also the [MEMory:STATe:RECall:AUTO](#) command.

| Command | Power on | *RST |
|------------------------------|---|--|
| *ESE | 0 | Not affected |
| *ESR | 0 | Not affected |
| *SRE | 0 | Not affected |
| *STB? | 0 | ??? |
| CAL[:MODE] | OFF | |
| CAL:STAT | ON if valid calibrating data for both voltage and current exists in the non-volatile memory, otherwise OFF. | |
| INST:NSEL | Power down state | Not affected |
| INST:SEL | Power down state | Not affected |
| OUTP[:STAT] | Power down state | OFF |
| [SOUR[n]]:CURR | Power down state | DEF (see Section 8.1) |
| [SOUR[n]]:CURR:PROT:DEL | Power down state | DEF (see Section 8.1) |
| [SOUR[n]]:CURR:PROT:STAT | Power down state | OFF |
| [SOUR[n]]:CURR:PROT:TRIP? | 0 | |
| [SOUR[n]]:CURR:STEP | Power down state | DEF (see Section 8.1) |
| [SOUR[<n>]]:LRIP:AUTO | Power down state | Not affected |
| [SOUR[n]]:POW:PROT[:LEV] | Power down state | DEF (see Section 8.1) |
| [SOUR[n]]:POW:PROT:DEL | Power down state | DEF (see Section 8.1) |
| [SOUR[n]]:POW:PROT:STAT | Power down state | ON |
| [SOUR[n]]:POW:PROT:TRIP? | 0 | |
| [SOUR[n]]:VOLT | Power down state | DEF (see Section 8.1) |
| [SOUR[<n>]]:VOLT:PROG[:SOUR] | INT | |
| [SOUR[n]]:VOLT:PROT:DEL | Power down state | DEF (see Section 8.1) |
| [SOUR[n]]:VOLT:PROT:STAT | Power down state | OFF |
| [SOUR[n]]:VOLT:PROT:TRIP? | 0 | |
| [SOUR[n]]:VOLT:STEP | Power down state | DEF (see Section 8.1) |
| [SOUR[<n>]]:VOLT:SENS[:SOUR] | Power down state | OFF |
| STAT:OPER[:EVEN] | 0 | Not affected |
| STAT:OPER:COND | 0 | Not affected |

| | | |
|----------------------------|------------------|--------------|
| STAT:OPER:ENAB | 0 | Not affected |
| STAT:OPER:INST[:EVEN] | 0 | Not affected |
| STAT:OPER:INST:COND | 0 | Not affected |
| STAT:OPER:INST:ENAB | 0 | Not affected |
| STAT:OPER:INST:ISUM[:EVEN] | 0 | Not affected |
| STAT:OPER:INST:ISUM:COND | 0 | Not affected |
| STAT:OPER:INST:ISUM:ENAB | 0 | Not affected |
| STAT:QUES[:EVEN] | 0 | Not affected |
| STAT:QUES:COND | 0 | Not affected |
| STAT:QUES:ENAB | 0 | Not affected |
| STAT:QUES:INST[:EVEN] | 0 | Not affected |
| STAT:QUES:INST:COND | 0 | Not affected |
| STAT:QUES:INST:ENAB | 0 | Not affected |
| STAT:QUES:INST:ISUM[:EVEN] | 0 | Not affected |
| STAT:QUES:INST:ISUM:COND | 0 | Not affected |
| STAT:OPER:INST:ISUM:ENAB | 0 | Not affected |
| SYST:ERR:COUN? | 0 | |
| SYST:POW | Power down state | ON |
| TEMP:PROT [MAIN] | Power down state | 70 |
| TEMP:PROT:DEL | Power down state | 10 |
| TEMP:PROT:STAT [MAIN] | Power down state | ON |

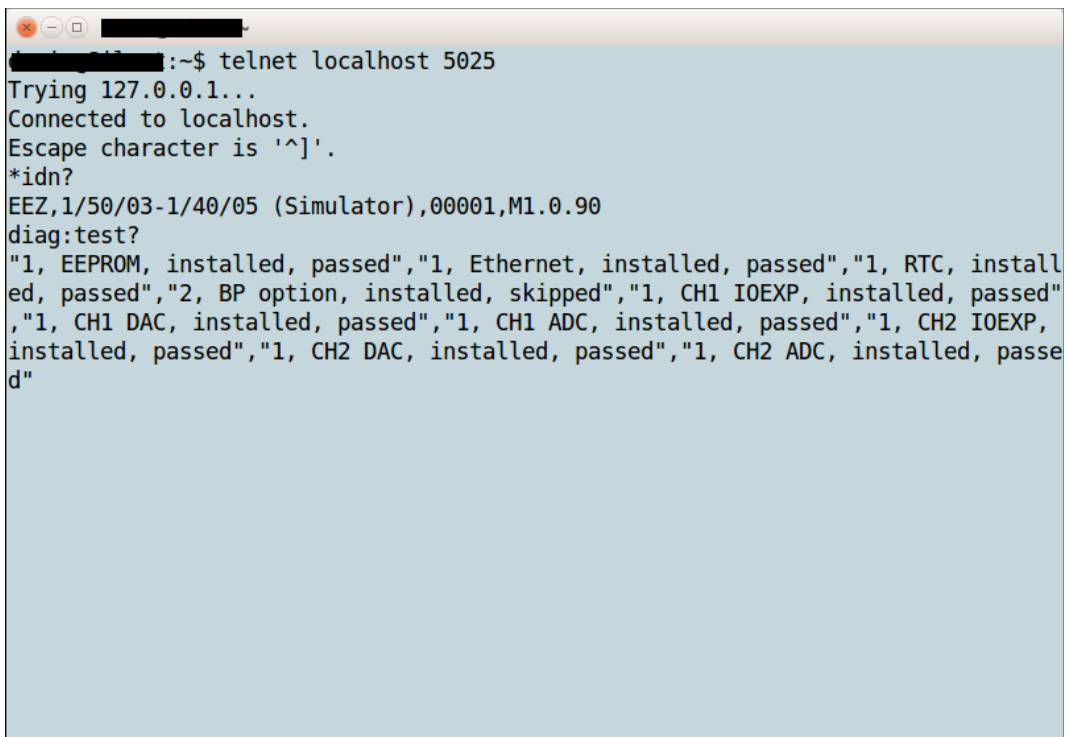
9. Software simulator

The PSU firmware can be also compiled and executed as a Windows, Linux or OS X application. The software simulator is a terminal application that can respond to any currently supported SCPI command described in this document. SCPI commands could be entered directly in the simulator's terminal window (Fig. 4) or remotely by using the e.g. a [Telnet](#) client (Fig. 5).

A terminal window titled "~/Downloads" with a light blue background. The text inside shows the command to run the simulator and its output.

```
~/Downloads$ ./eez_psu_sim
EEZ PSU software simulator ver. M1.0.90
Listening on port 5025
```

Fig. 4: Simulator welcome screen

A terminal window showing a telnet connection to the simulator. The text includes the telnet command, connection status, escape character, and a diagnostic test result.

```
telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
*idn?
EEZ,1/50/03-1/40/05 (Simulator),00001,M1.0.90
diag:test?
"1, EEPROM, installed, passed","1, Ethernet, installed, passed","1, RTC, installed, passed",
"2, BP option, installed, skipped","1, CH1 IOEXP, installed, passed",
"1, CH1 DAC, installed, passed","1, CH1 ADC, installed, passed","1, CH2 IOEXP, installed, passed",
"1, CH2 DAC, installed, passed","1, CH2 ADC, installed, passed"
"
```

Fig. 5: Remote connection using a telnet client

The simulator also has a GUI part when started open a separate window with the picture of the PSU front panel.

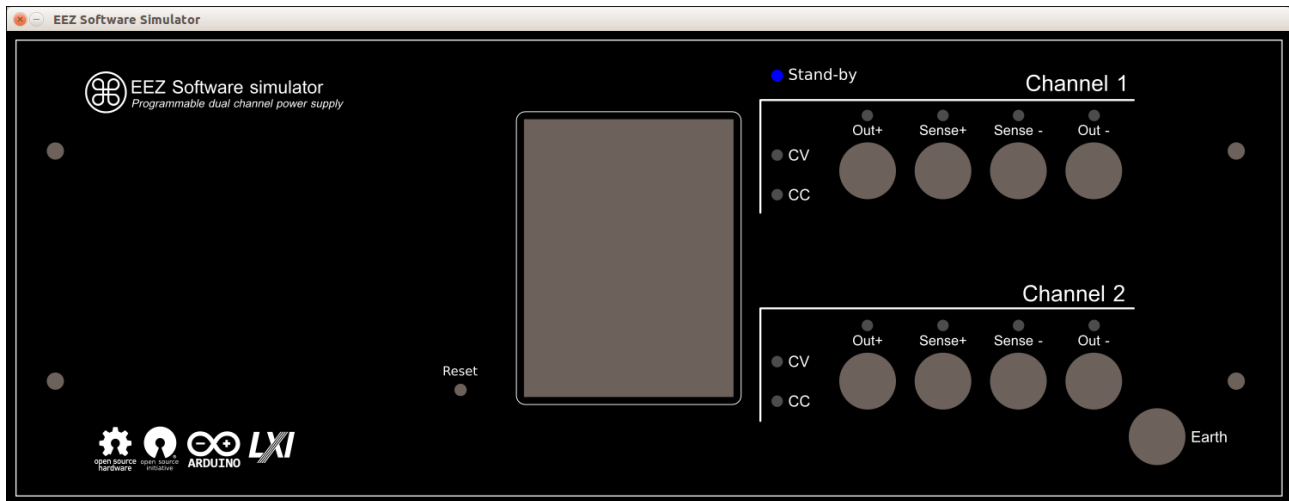


Fig. 6: Simulator GUI front panel

The GUI simulator front panel currently displays only changes in LED indicators and indicates if load is applied on the output terminals or not. For example if 8.2 Ohm load is connected to the channel 1 and cause that channel enters CC mode the GUI front panel will indicate that in the following way:

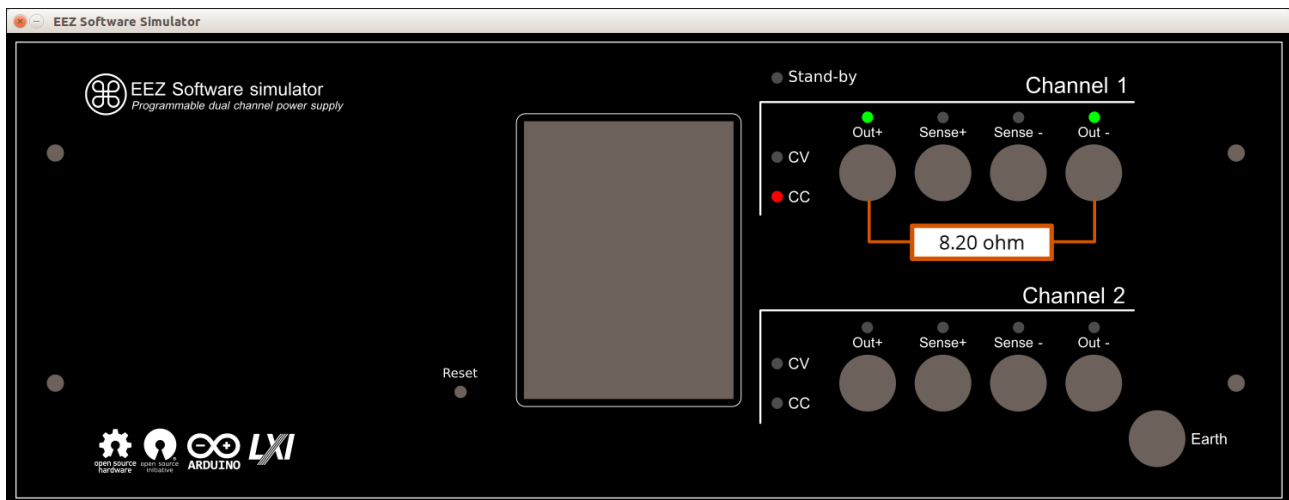


Fig. 7: Simulator GUI with connected load on the channel 1

9.1. SIMUlator

The SIMUlator subsystem represent a set of unclassified SCPI commands that can be used to manage external parameters and events such as load impedance, connection and disconnection of the load, sensor temperature or the PSU control circuit power supply state. Thanks to them e.g. simulation of the measuring and protection commands that depends of external events become more meaningful.

For example MEASure:CURRent? without connected load will always returns zero, or activation of the VOLTage:PROTection:STATe will automatically trip the OVP since channel cannot starts in CC mode of operation when output is switched on, etc.

| SCPI command | Description |
|---|---|
| SIMUlator | |
| :EXIT | Closes simulator |
| :GUI | Starts simulator's GUI |
| :LOAD {<value>} | Sets value of the virtual load |
| :STATe {<bool>} | "Connects" virtual load to the channel output |
| :PWRGood {<bool>} | Sets the PWRGOOD signal state |
| :TEMP {<value>} | Sets the temperature sensor value |

9.1.1. SIMUlator:EXIT

Syntax [SIMUlator:EXIT](#)

Description This command close all Software simulators windows (terminal and GUI if started).

Return None

Usage example SIMU:EXIT

Errors

Related Commands

9.1.2. SIMUlator:GUI

Syntax [SIMUlator:GUI](#)

Description Use this command to start the GUI simulator in the new window. See Fig. 6

Return None

Usage example SIMU:GUI

Errors

Related Commands

9.1.3. SIMUlator:LOAD

Syntax [SIMUlator:LOAD {<value>}](#)
[SIMUlator:LOAD?](#)

Description This command is used to define impedance of the virtual load that is connected to the channel output. Units are in ohms. With load connected it is possible to simulate e.g. CC mode of operation, current and power measurement, OCP and OPP functionality, etc.

The simulator currently cannot emulate the "UR" mode of operation (see the

[OUTPut:MODE?](#) command).

| Parameters | Name | Type | Range | Default |
|------------------|--|---------|----------------------|---------|
| | <value> | Numeric | 0 – 9999999 INFinite | – |
| Return | The query command returns the programmed load value. | | | |
| Usage example | SIMU:LOAD 10 | | | |
| Errors | | | | |
| Related Commands | OUTPut:MODE? | | | |

9.1.4. SIMULator:LOAD:STATe

| Syntax | SIMULator:LOAD:STATe {<bool>} SIMULator:LOAD? | | | |
|------------------|--|----------|------------|---------|
| Description | This command is used to “connect” or “disconnect” the virtual load to the channel output. If the GUI simulator is started (the SIMULator:GUI command) load symbol with selected value in Ohms will be displayed (See Fig. 7) | | | |
| Parameters | Name | Type | Range | Default |
| | <bool> | Discrete | ON OFF 0 1 | – |
| Return | The query command returns the load state. | | | |
| Usage example | SIMU:LOAD:STAT? | | | |
| | 0 | | | |
| | MEAS:CURRE? | | | |
| | 0.00 | | | |
| | SIMU:LOAD:STAT ON | | | |
| | MEAS:CURRE? | | | |
| | 1.50 | | | |
| Errors | | | | |
| Related Commands | OUTPut:MODE? SIMULator:GUI SIMULator:LOAD | | | |

9.1.5. SIMULator:PWRGood

| Syntax | SIMULator:PWRGood {<bool>} SIMULator:PWRGood? | | | |
|---------------|---|----------|------------|---------|
| Description | This command can be used to simulate detection of an internal power supply failure. When PWRGOOD signal is changed from 1 to 0 the PSU is going to the Stand-by mode (equal to the command SYSTem:POW OFF). The PSU mode cannot be changed until PWRGOOD is not changed to 1. | | | |
| Parameters | Name | Type | Range | Default |
| | <bool> | Discrete | ON OFF 0 1 | ON |
| Return | The query command returns the PWRGOOD signal state. | | | |
| Usage example | SYST:POW? | | | |
| | 1 | | | |
| | SIMU:PWRG 0 | | | |
| | SYST:POW? | | | |
| | 0 | | | |

10. Programming examples

10.1. Set channel output values and working with the OCP

This is a SCPI commands sequence that sets a voltage, current, and the over-current protection (OCP) on the channel two:

| | |
|---------------------|---|
| INST? | <i>Check currently selected output</i> |
| 1 | |
| INST CH2 | <i>Select channel two as current channel</i> |
| VOLT 10 | <i>Set output voltage</i> |
| CURR | <i>Set output current</i> |
| CURR:PROT:STAT? | <i>Check OCP status</i> |
| 0 | |
| CURR:PROT:STAT 1 | <i>Enable OCP</i> |
| CURR:PROT:DEL 100ms | <i>Set OCP delay</i> |
| OUTP 1 | <i>Enable output</i> |
| MEAS? | <i>Measure output voltage</i> |
| 10.00 | |
| MEAS:CURR? | <i>Measure output current</i> |
| 0.00 | <i>Current is zero since no load is connected</i> |

If [software simulator](#) is used, connection of the load can be also simulated:

| | |
|--------------|--|
| SIMU:LOAD 20 | <i>Define connected load impedance</i> |
| MEAS? | <i>Measure voltage once again</i> |
| 10.00 | |
| MEAS:CURR? | <i>Measure current once again</i> |
| 0.50 | <i>Measured current</i> |

The following command sequence could be used to test channel mode with load previously defined and after the load impedance is lowered enough that output current reach programmed value. The OCP has to be disabled because previously defined 100ms delay does not give us enough time to execute the whole sequence for testing channel mode and output voltage and current values:

| | |
|--------------------|---|
| OUTP:MODE? | <i>Check mode of operation</i> |
| "CV" | <i>The channel is in constant-voltage mode since output current is below previously programmed level</i> |
| SIMU:LOAD? | <i>Check load value</i> |
| 10 | |
| CURR:PROT:STAT? | <i>Check OCP status</i> |
| 1 | |
| CURR:PROT:STAT OFF | <i>Disable OCP</i> |
| SIMU:LOAD 4 | <i>Decrease load impedance</i> |
| OUTP:MODE? | <i>Check once again mode of operation</i> |
| "CC" | <i>Channel enters constant-current mode since $I_{max} = U / R = 10 / 4 = 2.5A$ and current is limited to the 1A</i> |

```

MEAS:CURREN?      Measure output current
1.00
MEAS:VOLT?        Measure output voltage
4.00              Output voltage is decreased since  $U = I * R = 1 * 4 = 4V$ 

```

The OCP will “trip” when output current reach programmed value and channel stay in the CC mode for more then programmed OCP delay time. To test that with e.g. the [software simulator](#) we'll disable channel output first, enable OCP and when change channel output back to enabled state:

```

OUTP OFF          Disable channel output
CURR:PROT:TRIP?   Check OCP status
0                OCP is not activated
CURR:PROT:STAT ON Enable OCP
VOLT?            Check programmed output voltage
10.00
CURR?            Check programmed output current
1.00
SIMU:LOAD?        Check simulated load value
4
OUTP ON           Enable channel output
CURR:PROT:TRIP?   Check OCP status once again
1                OCP has been tripped
OUTP?            Check channel output state
0                Channel output is changed to OFF stated by the OCP

```

The channel output state cannot be changed to enabled until any of protection is active. We have to clear protection first. If the same load that caused the first protection trip is still connected the channel output will be disabled immediately after the protection programmed delay time expired. Therefore we also have to disconnect load or disable protection. The later method will be used in the command sequence that follows:

```

OUTP ON           First attempt to enable channel output
OUTP?            0                This attempt failed, the channel output remain disabled
OUTP:PROT:CLEAR Channel protections reset
OUTP ON           Second to enable channel output
0                Channel output was enabled for a short time (100ms) and returns back to OFF state
CURR:PROT:TRIP?   Check OCP status
1                OCP has been tripped
OUTP:PROT:CLEAR Reset channel protections once again
CURR:PROT:STAT OFF Disable OCP
OUTP ON           Third attempt to enable channel output
OUTP?            1                Output is finally enabled
OUTP:MODE?

```

"CC"

Channel enters CC mode of operation

10.2. Voltage and current calibration

For optimum calibration results the following condition are recommended:

- the calibration ambient temperature is stable and between 20 °C and 30 °C.
- ambient relative humidity is less than 80%.
- Allow a 1-hour warm-up period before verification or calibration (use DIAG:OTIM? LAST).
- Use short and thick cables to connect test setups.

| Step | Commands | Description |
|------|---|---|
| 1 | INST {CH1 CH2}; OUTP ON | Select the channel to be calibrated and enable the channel output . |
| 2 | VOLT:PROT:STAT OFF POW:PROT:STAT OFF | Disable if required the voltage and power protection function. |
| 3 | CAL ON, "<password>" | PSU enters calibration mode on the channel selected in step 1. Both voltage and current on the selected channel are set to the MINimum value. The VOLT? and CURR? commands can be optionally used here to test channel output values. |
| 4 | | For voltage calibration, connect a digital voltmeter (DVM) across the PSU's output terminals. |
| 5 | CAL:VOLT:LEV MIN | Set the channel to the low-end (MIN) calibration point. |
| 6 | CAL:VOLT 0.249 | Enter the reading you obtained from the external DVM. |
| 7 | CAL:VOLT:LEV MID | Set the channel to the middle (MID) calibration point. |
| 8 | CAL:VOLT 25.058 | Enter the reading you obtained from the DVM. |
| 9 | CAL:VOLT:LEV MAX | Set the channel to the high (MAX) calibration point. |
| 10 | CAL:VOLT 49.86 | Enter the reading you obtained from the DVM. |
| 11 | | For current calibration, connect an appropriate current monitoring resistor (shunt) across the output terminals and connect the DVM across the shunt resistor. |
| 12 | | Repeat step 5 through step 10 by substituting CURR for VOLT for current calibration. For example, CAL:CURR:LEV MIN. The OCP function has to be disabled (CURR:PROT:STAT OFF). |
| 13 | | Repeat step 1 through step 12 for the other channel calibration. |
| 14 | CAL:REM "<string>" | Record calibration information such as next calibration due date for future reference. The calibration string may contain up to 40 characters. |
| 15 | CAL:SAVE | Save to non-volatile memory new calibration data. |
| 16 | CAL OFF, "<password>" | PSU exit calibration mode. Both voltage and current on the selected channel are again set to the MINimum value. |

10.3. Working with profiles

The following command sequence could be used to store current set of parameters to the profile location 4 in the non-volatile memory:

MEM:STAT:VAL? 4 Check to see if profile selected location is empty

0

MEM:STAT:NAME? 4 We can also check that by querying profile location name

"--Not used--"

| | |
|---|--|
| INST CH1 VOLT?;:CURR?;:OUTP? 0.00;0.00;0 | <i>Examine currently programmed output values of the first channel</i> |
| INST CH2 VOLT?;:CURR?;:OUTP? 0.00;0.00;0 | <i>Examine currently programmed output values of the second channel</i> |
| VOLT 12;:CURR 300mA INST CH1 VOLT 12;:CURR 300mA OUTP 1;:OUTP 1, CH2 | <i>Reprogram both channel output values that will be stored as a new profile</i> |
| *SAV 4 | <i>All profile parameters is now storing on the selected location</i> |
| MEM:STAT:NAME? 4 "" | <i>Check profile name</i> |
| MEM:STAT:NAME 4, "Dual 12V/300mA, Output ON" | <i>Set the profile name (only ASCII characters are allowed!)</i> |
| MEM:STAT:NAME? 4 "Dual 12V/300ma, Output ON" | <i>Check the profile name once again</i> |

We can now turn the PSU off (when it enters the Stand-by mode) turn it on again and check some of the programmed parameters:

| | |
|---|--|
| SYST:POW 0 | <i>The PSU enters the Stand-by mode</i> |
| SYST:POW 1 | <i>Returns back from the Stand-by mode. Please note that this command can be executed with the minimum of 5 seconds delay otherwise a -200,"Execution error" will be generated (you can check that with the SYST:ERR? command)</i> |
| VOLT?;:CURR?;:OUTP? 0.00;0.00;0 | <i>Query programmed voltage, current and output state of the currently selected channel Returned data indicate that previously saved values in profile number 4 were not used</i> |
| *RCL 4 VOLT?;:CURR?;:OUTP? 12.00;0.30;1 | <i>Recall parameters from desired location and execute query once again The channel output values are now programmed using the selected profile</i> |

We can automate above mentioned process that channel profile parameters stored in non-volatile memory are using on power up. First we'll check what is a current status of automatic recall and what profile will be used in the case of automatic recall:

| | |
|-------------------------|---|
| MEM:STAT:REC:AUTO? 0 | <i>Query status of automatic profile recall during power on sequence Automatic recall is turned off</i> |
| MEM:STAT:REC:AUTO ON | <i>Turn on automatic recall</i> |
| MEM:STAT:REC:SEL? 0 | <i>Query which profile will be used when automatic recall is turned on Selected profile was 0</i> |
| MEM:STAT:REC:SEL 4 | <i>Change power on profile to 4</i> |
| SYST:POW 0 | <i>Switch the PSU to the Stand-by mode once again</i> |
| SYST:POW 1 | <i>Returns back from the Stand-by mode. Again wait at least 5 sec-</i> |

| | |
|---------------------|---|
| | <i>ends before enters this command</i> |
| VOLT?;:CURR?;:OUTP? | <i>Query programmed voltage, current and output state of the currently selected channel</i> |
| 12.00;0.30;1 | <i>The channel output values are programmed using the selected profile</i> |

10.4. Get identification info and self-test results

The PSU's identification information could be beneficial when more than one instrument are controlled. Additionally in the following example information about self-test will be queried:

| | |
|---|--|
| *IDN? | <i>Query identification string</i> |
| EEZ,1/50/03-1/40/05 (Due),00001,M1.0.93 | <i>PSU with two different channels is identified, the first channel is 0-50V/3A and the second is 0-40V/5A. Serial number is 00001, and firmware version M1.0.93</i> |
| *TST? | <i>Execute self-test and query result</i> |
| 0 | <i>Self-test is passed</i> |
| DIAG:TEST? | <i>Query additional information about self-test</i> |
| "1, EEPROM, installed, passed", "1, Ethernet, installed, passed", "1, RTC, installed, passed", "1, DateTime, installed, passed", "2, BP option, installed, skipped", "1, CH1 IOEXP, installed, passed", "1, CH1 DAC, installed, passed", "1, CH1 ADC, installed, passed", "1, CH2 IOEXP, installed, passed", "1, CH2 DAC, installed, passed", "1, CH2 ADC, installed, passed" | |

The self-test could be performed even when the PSU is in the Stand-by mode. We'll first switch the PSU into the Stand-by mode. At the end of this example we are using additional diagnostic command that allows us to query information about channel's ADC measurements.

| | |
|---|--|
| SYST:POW 0 | <i>The PSU enters the Stand-by mode</i> |
| DIAG:TEST? | <i>Query additional information about self-test</i> |
| "1, EEPROM, installed, passed", "1, Ethernet, installed, passed", "1, RTC, installed, passed", "1, DateTime, installed, passed", "2, BP option, installed, skipped" | <i>Only Arduino Shield +BP module diagnostic information is returned</i> |
| SYST:POW 1 | <i>Returns back from the Stand-by mode</i> |
| DIAG:ADC? | <i>Additional information about currently selected channel ADC inputs</i> |
| "U_SET=12.02 V", "U_MON=12.00 V", "I_SET=0.30 A", "I_MON=0.00 A" | <i>U_SET and I_SET are measured values of the DAC outputs, U_MON and I_MON are actual output values. I_MON is 0 because no load is connected. A small difference between set and actual output voltage exists because calibration data are currently in use.</i> |

11. SCPI commands scheduled for the Milestone Three (M3)

Please note that the following list is preliminary and more accurate lists will be available after the Milestone Two (M2) is reached.

| SCPI command | Description |
|--|--|
| DCL | Requires the client (controller) to send a "DCL\n" string |
| *SRE | Sets the value of the Service Request Enable register |
| *TRG | Generates a software trigger |
| *WAI | Waits until all pending commands are completed |
| ABORT | Resets the trigger system and places all trigger sequences in the IDLE state |
| CALibration | |
| :TEMPerature | |
| [:DATA] {<new value>} | Enters the calibration value |
| DIAGnostic | |
| [:INFormation] | |
| :BOARDs? | Returns a list of the installed boards in the PSU |
| :CCOunt:PON? | Returns the cumulative number of power-ons |
| :CCOunt:PROTection:TEMPerature? {<sensor>} | Returns the cumulative number of OTP activation |
| :FIRMware? | Returns the firmware information |
| DISPlay {<bool>} | Sets the front panel TFT display state |
| :BRIGhtness {<numeric_value>} | Sets the intensity of the front panel TFT display |
| :TEXT {<quoted string>} | Displays a message on the front panel TFT display |
| :CLEar | Clear a message on the front panel TFT display |
| :CMAP | |
| :DEFault | Resets all color maps to default value |
| :COLor | |
| :RGB {<red>,<green>,<blue>} | Sets the RGB values for the color map |
| INITiate | |
| [:IMMediate] | Completes one full trigger cycle |
| :CONTInuous {<bool>} | Enables/disables continuous transient triggers |
| INSTrument | |
| :COUPle | |
| [:TRIGger] | Trigger coupling between two logical outputs |
| OUTPut | |
| :COUPle | |

| | |
|-------------------------------------|--|
| [:STATe] {<bool>} | Enables channel coupling for output synchronization |
| :INSTrument | Selects which channels are coupled |
| :DPRog {<bool>} | Disables down-programmer circuit |
| :COUPle {<bool>} | Enables channel coupling for protection faults |
| :TRACk[:STATe] {<bool>} | Enables tracking mode |
| [SOURce[<n>]] | |
| :CURRent | |
| [:LEVel] | |
| :TRIGgered [:AMPLitude] {<current>} | Sets the triggered output current |
| :MODE {<mode>} | Sets the current trigger mode |
| :LIST | |
| :COUNT | Sets the number of times that the list is executed |
| :CURRent[:LEVel] | Specifies the current setting for each list step |
| :DWELl | Specifies the dwell time for each list step |
| :LOAD {<list_number>} | Loads stored list from the non-volatile memory |
| :SAVE {<list_number>} | Saves LIST to the non-volatile memory |
| :VOLTage[:LEVel] | Specifies the voltage setting for each list step |
| :LOCK | Channel parameters are locked and cannot be changed without providing password |
| :VOLTage | |
| [:LEVel] | |
| :TRIGgered [:AMPLitude] {<voltage>} | Sets the triggered output voltage |
| :MODE {<mode>} | Sets the voltage trigger mode |
| SYSTEM | |
| :COMMunicate | |
| :ETHernet | Ethernet communication parameters |
| :CONTRol? | Queries communication port for SRQ handling |
| :DHCP {<bool>} | Enables the DHCP mode |
| :DNS {<ip_address>} | Sets the DNS (Domain Name Service) address |
| :GATEway {<ip_address>} | Sets the network gateway address |
| :HOSTname {<name>} | Sets the Ethernet communication host name |
| :IPAdDress {<ip_address>} | Sets the IP address |
| :MAC {<mac_address>} | Returns the Ethernet device MAC address |
| :PORT {<number>} | Sets the Ethernet communication port |
| :SMASK {<mask>} | Sets the subnet mask |
| :NTP {<ip_address>} | Sets NTP (Network Time Protocol) server |
| :SERial | Serial (via USB) communication parameters |
| :BAUD {<speed>} | Sets the baud rate |

11. SCPI commands scheduled for the Milestone Three (M3)

| | |
|--|--|
| :BITS {<number>} | Sets the number of data bits |
| :PARity {<parity>} | Sets parity bit |
| SBITs {<number>} | Sets the number of stop bits |
| :ERRor | |
| :ALL? | Queries the error/event queue for all the un-read items |
| :CODE[:NEXT]? | Queries the error/event queue for the next item code |
| :GROup[:DEFine] | |
| :PARAllel | Makes single output from multiple channels connected in parallel |
| :SERial | Makes single output from multiple channels connected in serial |
| :DELete[:ALL] | Deletes any existing group of channels |
| :KLOCK | Lock the local input device |
| :LFRequency {<value>} | Sets power-line frequency |
| :LOCAL | Returns from remote mode to local mode |
| :PASSword:NEW {<old code>, <new code>} | Changes system password |
| :REMote | Places the PSU in the remote mode of operation |

12. SCPI commands summary

| Common command | Description |
|--|--|
| *CLS | Clears all status data structures |
| *ESE {<value>} | Programs the Standard Event Status Enable register bits |
| *ESR? | Reads the Standard Event Status Register |
| *IDN? | Returns the UNIQUE identification of the PSU |
| *OPC | Operation Complete Command used for program synchronization |
| *RCL {<profile>} | Recalls the PSU state stored in the specified storage location |
| *RST | Reset PSU to the initial state |
| *SAV {<profile>} | Stores the current PSU state in the specified storage location |
| *STB? | Reads the Status Byte register |
| *TST? | Returns Self-Test results |

| SCPI Command | Description |
|---|--|
| CALibrate[:MODE] {<bool>, <password>} | Enables/disables calibration mode |
| :CLEar {<password>} | Clears all calibration parameters |
| :CURRent | |
| [:DATA] {<new value>} | Enters the calibration value |
| :LEVel {<level>} | Calibrates the output current programming |
| :PASSword | |
| :NEW {<old>, <new>} | Changes calibration password |
| :REMark {<string>} | Saves calibration information |
| :SAVE | Saves the new cal constants in non-volatile memory |
| :STATe {<bool>, <password>} | Enables calibration parameters |
| :VOLTage | |
| [:DATA] {<new value>} | Enters the calibration value |
| :LEVel {<level>} | Calibrates the output voltage programming |

| | |
|--------------------------------|--|
| DIAGnostic | |
| [:INFormation] | |
| :ADC? | Returns the latest values acquired by ADC |
| :CALibration? | Returns a list of the calibration parameters |
| :PROTection? | |
| :TEST? | Returns results of the most recent self-test |

| | |
|---|-------------------------------------|
| INSTrument | |
| [:SElect] {<channel>} | Selects the output to be programmed |
| :NSElect {<channel_number>} | Selects the output to be programmed |

| | |
|--|--|
| MEASure | |
| [:SCALar] | |
| :CURRent | |
| [:DC]? {<channel>} | Takes a measurement; returns the average current |

| | |
|----------------------------|--|
| :POWer | |
| [:DC]? [<channel>] | Takes a measurement; returns the average power |
| :TEMPerature | |
| [:DC]? [<sensor>] | Takes a measurement; returns the average temperature |
| [:VOLTage] | |
| [:DC]? [<channel>] | Takes a measurement; returns the average voltage |
| <hr/> | |
| MEMory | |
| :NSTates? | Returns total number of state storage memory locations |
| :STATe | |
| :CATalog? | Lists the names associated with all ten state storage locations |
| :DELeTe [<profile>] | Deletes the contents of a state storage location |
| :ALL | Deletes the contents of all state storage locations |
| :NAME [<profile>, <name>] | Assigns a custom name to a state storage locations |
| :RECall | |
| :AUTO [<bool>] | Specifies whether the power-down state is recalled from location 0 on power-on |
| :SELeCt [<profile>] | Specifies which PSU state will be used at power on |
| :VALid? [<profile>] | Determines whether a storage location contains a valid state |
| <hr/> | |
| OUTPut | |
| [:STATe] [<bool>] | Enables the specified output channel(s) |
| :MODE? | Returns the channel mode of operation |
| :PROTection | |
| :CLEAr | Resets latched protection |
| :SENSe | Enables the remote sense function of the channel |
| <hr/> | |
| SIMUlator | |
| :EXIT | Closes simulator |
| :GUI | Starts simulator's GUI |
| :LOAD [<value>] | Sets value of the virtual load |
| :STATe [<bool>] | "Connects" virtual load to the channel output |
| :PWRGood [<bool>] | Sets the PWRGOOD signal state |
| :TEMP [<value>] | Sets the temperature sensor value |
| <hr/> | |
| [SOURce[<n>]] | |
| :CURRent | |
| [:LEVel] | |
| [:IMMediate][:AMPLitude] | |
| :<current> | Sets the output current |
| :STEP[:INCRement] [<step>] | Sets the step of the current change |
| :PROTection | |
| :DELay | |

| | |
|--|--|
| [:TIME] {<time>} | Sets the over-current protection (OCP) programming delay |
| :STATe {<bool>} | Enables/disables over-current protection on the selected channel |
| :TRIPped? | Returns status of over-current protection activation |
| :POWer | |
| :PROTection[:LEVel] | Sets the over-power protection (OPP) level |
| :DELay | |
| [:TIME] {<time>} | Sets the over-power protection programming delay |
| :STATe {<bool>} | Enables/disables over-power protection on the selected channel |
| :TRIPped? | Returns status of over-power protection activation |
| :VOLTage | |
| [:LEVel] | |
| [:IMMediate][:AMPLitude] {<voltage>} | Sets the output voltage |
| :STEP[:INCRement] {<step>} | Sets the step of the voltage change |
| :PROTection | |
| :DELay | |
| [:TIME] {<time>} | Sets the over-voltage protection (OVP) programming delay |
| :STATe {<bool>} | Enables/disables over-voltage protection on the selected channel |
| :TRIPped? | Returns status of over-voltage protection activation |

STATus

| | |
|---|---|
| :OPERation | |
| [:EVENT]? | Returns the value of the Operation Event register |
| :CONDition? | Returns the value of the Operation Instrument Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Event register |
| :INSTrument[<n>] | |
| [:EVENT]? | Returns the value of the Operation Instrument Event register |
| :CONDition? | Returns the value of the Operation Instrument Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Instrument Event register |
| :ISUMmary<n> | |
| [:EVENT]? | Returns the value of the Operation Instrument Isummary Event register |
| :CONDition? | Returns the value of the Operation Instrument Isummary Condition register |
| :ENABLE {<value>} | Enables specific bits in the Operation Instrument Isummary Event register |
| :PREset | Presets all enable registers to power-on state |
| :QUEStionable | |

| | |
|---|--|
| [:EVENT]? | Returns the value of the Questionable Event register |
| :CONDition? | Returns the value of the Questionable Condition register |
| :ENABle {<value>} | Enables specific bits in the Questionable Event register |
| :INSTrument[<n>] | |
| [:EVENT]? | Returns the value of the Questionable Instrument Event register |
| :CONDition? | Returns the value of the Questionable Instrument Condition register |
| :ENABle {<value>} | Enables specific bits in the Questionable Instrument Event register |
| :ISUMmary<n> | |
| [:EVENT]? | Returns the value of the Questionable Instrument Isummary Event register |
| :CONDition? | Returns the value of the Questionable Instrument Isummary Condition register |
| :ENABle {<value>} | Enables specific bits in the Questionable Instrument Isummary Event register |

SYSTem

| | |
|--|---|
| :BEEPer[:IMMEDIATE] | Issues a single beep immediately |
| :STATe {<bool>} | Enables beeper function |
| :CAPability? | Returns an <instrument_specifier> |
| :DATE {<yyyy>,<mm>,<dd>} | Sets the date of the system clock |
| :ERRor | |
| [:NEXT]? | Queries and clears errors from the error queue |
| :COUNT? | Queries the error/event queue for the number of unread items |
| :POWER {<bool>} | Enters the PSU into the Stand-by mode |
| :TEMPerature | |
| :PROTection | |
| [:HIGH] | |
| [:LEVel] {<temperature>[, <sensor>]} | Sets the OTP value |
| :CLEAr[, {<sensor>}] | Clears the latched protection status of the over-temperature protection (OTP) |
| :DELay | Sets time-out period |
| [:TIME] {<delay>[, <sensor>]} | Sets the OTP programming delay |
| :STATe {<bool>[, <sensor>]} | Enables/disables OTP on the selected temperature sensor |
| :TRIPped? [<sensor>] | Returns status of OTP activation |
| :TIME {<hh>,<mm>,<ss>} | Sets the time of the system clock |
| :VERSIon? | Returns the SCPI version number |

