TET Electronics IndustrieAlpine GmbH & Co.KG
IndustrieAlpine Allee 1
D-94513 Schönberg
Tel.: (49) 85 54 / 96 09-0
Fax: (49) 85 54 / 96 09 20
Mail: sales@tetelectronics.de

Operation Manual
Option 34/1

Interface IEEE / RS232

Contents

1 Hints

1.1 System

The Option 34 is an interface card, which enables remote control of voltage, current and overvoltage protection for TET power supplies. Option 34 cards include RS232 and optionally a GPIB interface. Furthermore the momentary values of voltage, current and of the programmable overvoltage protection can be monitored.

The power supply supports SCPI in version 1999.0 (Standard Commands for Programmable Instruments). Most instrument manufacturers comply with SCPI. The SCPI standard specifies a command structure and syntax for programmable instruments control. SCPI incorporates the IEEE488.2 standard. Moreover the programmer benefits from the unified error handling and status reporting. Chapter 4 presents an introduction to SCPI and the according status registers. A complete list of commands can be found in Chapter 5.

1.2 Choosing the programming syntax

If you like to use the formerly used TET commands in contrast to SCPI, then set jumper J4 (see Fig. 1). The commands are TET-specific and don't comply with the SCPI and IEEE 488.2 standard.

In case of existing instrumentation software, which was written using the old TET commands and because rewriting the software to SCPI would consume too much time and effort, you can still use the old TET-specific commands.

Please note that SCPI- and TET-commands can not be used simultaneously. Jumper 4 configures the syntax, which is used by the Option 34 card:

J4 open:	SCPI-Syntax
J4 closed:	TET-Syntax

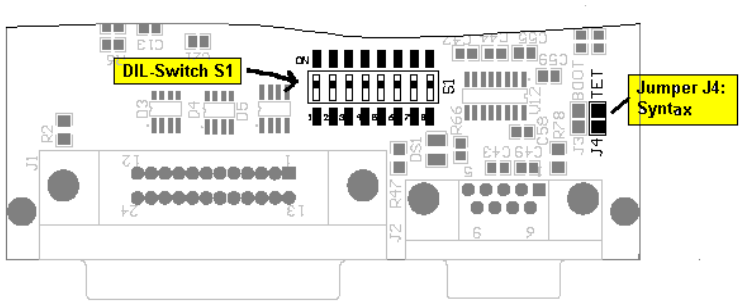


Fig. 1: arrangement of jumper J4 and DIL-switch S1

1.3 Switching to remote control

After switching on the power supply the display indicates the IEEE488-address for about 2 seconds, followed by rated voltage and rated current. The power supply is ready for use then and indicates the actual output voltage and current.

Please note that the device is in manual mode („local mode“) after switch-on and can be run by the front panel pots and knobs.

Switching to remote control („remote mode“) is done

- automatically with an active GPIB connection, if the device receives an addressed command by a talker, usually the bus controller.
- with active RS232 interface, if the device receives the custom command @REM (respectively command “B1” if TET-syntax is used).
- manually with the button “INT/EXT”, which is located on the front panel of the power supply.

After changing to remote mode the power supply is set to:

- output voltage = 0 V
- current limit = 0 A
- overvoltage protection (OVP) = 120% of rated voltage

The power supply remains in remote mode until it is switched back to local mode either manually by “INT/EXT” switch or by remote control (e.g. with the command @LOC).

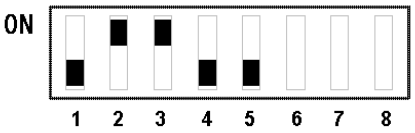
2 IEEE488 Interface

The configuration of IEEE488 and RS232 parameters is done with DIL-switch S1, which is located on the Option 34 board (see Fig.1).

2.1 IEEE488 – Setting the device address

The IEEE488 device address is configured with digits 1 to 5 of the DIL switch. Please note that the familiar binary system is used:

DIL-switch	1	2	3	4	5
value	1	2	4	8	16



Example: DIL-switch 2 and 3 are set to ON in order to get device address 6.

The **factory setting** of the device address is: **2**.

Valid IEEE488 addresses are in the range of 0 to 30, whereas address 0 is commonly used by bus controller cards. Address 31 is not allowed to be used by devices.

2.2 IEEE488 – Message Termination

Basically GPIB data transfers can be terminated as follows:

(1) The GPIB includes a hardware line (EOI) that can be asserted with the last data byte.

(2) A specific end-of-string (EOS) character is placed at the end of the data string itself. Some instruments use this method instead of or in addition to the EOI line assertion.

For maximum flexibility you can choose between 9 different message terminations.

Termination code “EOI + LF” is factory setting. At the end of the data string the EOS-character “LF” (0xA) is attached and simultaneously the hardware line EOI asserted. This is the preferred behavior in IEEE488.2 systems.

2.2.1 EOI (End or Identify)

If message termination shall be done solely by EOI, then DIL-switch 6 is set to position ON and DIL-switch 8 to position OFF.

In the case, that another termination is necessary, then you can set DIL-switch 6 and 8 to OFF positon (see Table 1). The chosen differing termination code has to be programmed into the power supply (see chapter 2.2.2).

setting of DIL-switch	termination code
	EOI
	Programmed termination code (see 2.2.2)

Table 1: Selection of EOI and programmed termination code

2.2.2 Programmed termination code

To proram a termination code, which varies from the simple EOI, following operations have to be completed:

1. Set DIL-switch 1 to 5 to address 31, which is not allowed for GPIB instruments. Choose the desired termination code with DIL-switch 6 to 8, according to table 2.
2. Switch-on the power supply and wait for the REM-LED blinking in a half second cycle. This shows that the desired termination code was saved in the internal flash memory of the power supply.
3. Switch-off the power supply, adjust an address from 1 to 30 (see chapter 2.1) and set DIL-switch 6 and 8 to OFF position. Henceforth the programmed termination code will be used.

setting of DIL-switch	programmable termination code
	EOI + CR
	EOI + CR + LF
	EOI + LF
	EOI + LF + CR
	CR

	CR + LF
	LF
	LF + CR

Table 2: programmable termination codes

2.3 SRQ (Service Request)

If you are using the TET-specific programming syntax, then you can allow a service request (SRQ) with DIL-switch 7. See table 3.

setting of DIL-switch	
	SRQ allowed
	SRQ not allowed

Table 3: setting of SRQ when using TET-specific syntax

If you are using the SCPI programming language, then it does not matter if DIL-switch 7 is OFF or ON. According to IEEE 488.2 and therefore SCPI the creation of SRQ is masked by register SRE (Service Request Enable) not by external DIL switches.

2.4 IEEE 488 capabilities

SH1	- source handshake
AH1	- acceptor handshake
T5	- basic talker, serial poll, unadressed on MLA, send END or EOS
L3	- basic listener, unadressed on MTA, detect END or EOS
SR1	- service request
RL1	- remote-/local
PP1	- remote parallel poll
DC1	- device clear
DT0	- no trigger capability
C0	- no controller capability

3 RS232 Interface

3.1 Wiring

For serial communication you need a cable with signals RxD, TxD and GND. Hardware handshake is not supported. Adequate is a standard RS232 cable with 1:1 wiring.

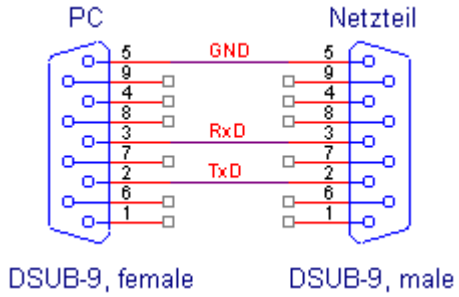


Bild 2: serial cable

3.2 RS232 interface parameter

The RS232 interface is set to the following parameters on delivery:

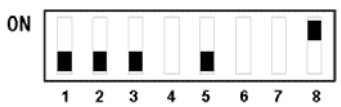
- 9600 baud
- 1 start bit
- 8 data bits
- 1 stop bit
- no parity
- termination code = terminal

With termination code „terminal“ you can use a software like „HyperTerminal“ or else to control the power supply with a remote computer.

Adjusting other RS232 interface parameters

To change the RS232 parameters you have to do a reprogramming by using the DIL-switch S1:

1. Enable the reprogramming sequence with DIL-switch 5 to OFF and 8 to ON.
2. Adjust the required baud rate with DIL-switch 1 to 3 according to table 4.
3. Choose the required serial frame with DIL-switch 4 according to table 5.
4. Select the required termination code with DIL-switch 6 and 7 according to table 6.
5. Switch on the device and wait until the remote LED flashes on and off each 0.5 seconds. The parameters are stored now.
6. Switch off the device and set DIL-switch 8 to OFF. If you are using also IEEE488 then choose the required GPIB address according to chapter 2.1.
7. Hence the power supply will use the programmed parameters after switching on again.

DIL switch setting	baud rate
	1200

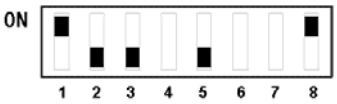
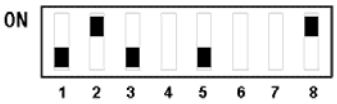
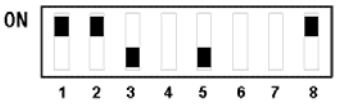
DIL switch setting	baud rate
	2400
	4800
	9600
	19200
	38400
	57600
	115200

Table 4: adjusting the baud rate

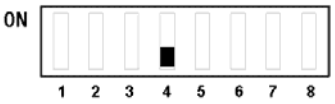
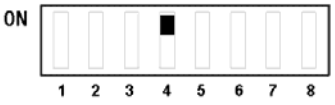
DIL switch setting	frame
	1 start bit, 8 data bits, 1 stop bit no parity
	1 start bit, 8 data bits, 1 stop bit, even parity

Table 5: choosing the serial frame

DIL switch setting	RS232 termination code
	LF (0x0A)
	CR (0x0D)
	^Z (Ctrl-Z) (0x1A)
	terminal (CR/LF, CR sent by terminal)

Table 6: choosing the RS232 termination code

4 Remote control with SCPI

The power supply complies with the SCPI version 1999.0 (Standard Commands for Programmable Instruments). SCPI specifies a programming language which is used to control automatic test equipment like power supplies, function generators, voltage and amp meters and so on.

SCPI is based on the standard IEEE 488.2. IEEE 488.2 specifies the transmission protocol and the so called “common commands”, which are supported by all 488.2 compliant devices. However SCPI defines additional commands, which are essential for the different category of devices, for example power supplies, function generators etc.

When using SCPI compliant instruments you can replace devices or even use different device manufacturers without the need of rewriting your control software right from the start. Furthermore there is no more need to learn a new command set for each newly acquired instrument, because you have one language for all instruments. This benefit helps to save time and costs.

4.1 Command syntax

SCPI is an upgrading of the 488.2 specification regarding to the format of data, to the application of „common commands“ and to the 488.2 status system. Just like with 488.2 there are “program messages”, which are sent from the bus controller to the instrument, and according “response messages”, sent from the instrument to the controller.

According to IEEE 488.2 SCPI specifies both commands and queries. For each command, e.g. adjusting a voltage, you will find the appropriate query to sense or read back the value. Exceptions are described in the command documentation.

4.1.1 „Common Commands“-Syntax

„Common commands“ are device-independent commands. They are realized by all 488.2-devices, and thus by all SCPI devices. „Common commands“ consist of a header with a `'*'` put in front and optionally a following parameter.

Examples:

*RST	reset device
*SRE 48	bits 4 and 5 of the „service request enable“ register are set
*SRE #H30	as *SRE 48 (48decimal = 30hex)
*SRE #B00110000	as *SRE 48 (48decimal = 00110000bin)
*SRE?	SRE register query

4.1.2 SCPI-Syntax

SCPI arranges the commands into different categories, the so called „sub systems“. Sub system SOURCE, for example, features commands with which an instrument can source physical values, generally speaking. These are voltage and electrical current regarding a power supply. For a waveform generator the SOURCE sub system defines additionaly commands regarding waveform and frequency of the signal.

tree structure

The commands of a sub system are arranged in a hierarchical structure similiarly the hierarchical file system of a personal computer.

Figure 3 shows the the SOURCE sub system of the TET power supply.

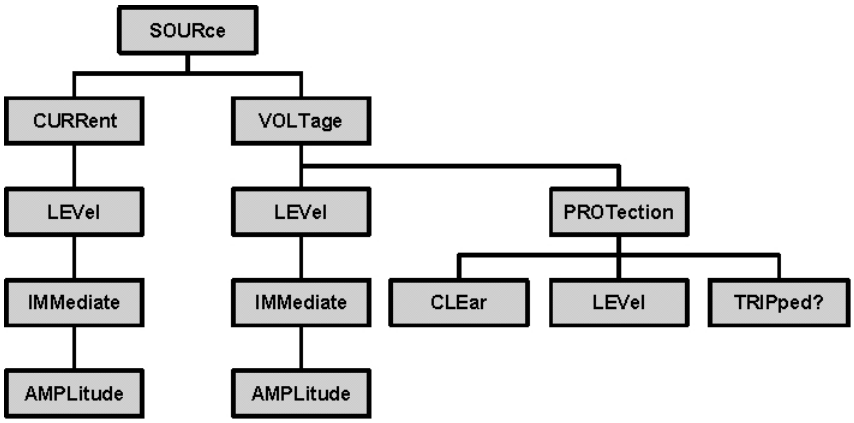


Fig. 3: command tree of the sub system SOURCE

Like in a file system the top level of the sub system is called ”root”. Commands arranged in a lower level have to be written using the entire path. To separate different levels a colon `':'` is used.

Wie bei einem Dateisystem wird die höchste Ebene als „root“ be-zeichnet. Befehle auf niedrigerer Ebene werden mit dem ganzen Pfad angegeben. Als Trennzeichen der Ebenen dient ein Doppelpunkt `':'`.

For example the command

```
SOURce:VOLTage:LEVel:IMMediate:AMPLitude 100V
```

sets the output of the power supply to 100 Volts.

path

- A message terminator sets the active path to „root“. Mostly <LF>, EOI or <LF> + EOI are used representing the message terminator.
- After power-on or after sending a *RST command the path is also set back to „root“.
- A colon (':') separates the levels of the hierarchy. Each time the command parser finds a colon within the command string, he moves down the hierarchy by one step.
- A semicolon separates two commands inside the same command string without changing the path.

Example:

```
SOURce:VOLTage:PROTection:CLEar; LEVel 48
```

is the same like

```
SOURce:VOLTage:PROTection:CLEar
SOURce:VOLTage:PROTection:LEVel 48
```

optional key words

Key words, which are enclosed in squared brackets, are optional and can be omitted. Command lines are shortening considerably if you are leaving aside optional key words. Optional key words are supported due to compliance to the SCPI standard.

Example: The command for adjusting an electrical current is defined by the SOURce sub system as follows (see also chapter 5):

```
[SOURce:]CURRent[:LEVel]
      [:IMMediate][:AMPLitude] <numeric_value>
```

To emit a current of 10 amps (respectively to set the current limit) you can use the following command:

```
SOURce:CURRent:LEVel:IMMediate:AMPLitude 10A
```

Or quickly, by leaving aside the key words in squared brackets:

```
CURRent 10A
```

long- and short form

Most of the key words are written like strings with upper-case letters followed by lower-case letters. The upper-case letters are marking the short form of the command. The lower-case letters are marking the long form of the command and can be left aside, if required.

For example, the command

```
SOURce:CURRent:LEVel:IMMediate:AMPLitude 10A
```

is the same as

```
SOUR:CURR:LEV:IMM:AMPL 10A
```

The upper and lower case is simply marking the long and short form of commands. SCPI itself does not distinguish between upper and lower case, for example:

`CURRent 10A` = `CURR 10A` = `curr 10A` = `current 10A`

parameter

- Whitespace characters like <tab> or <space> are ignored by the command parser. However, whitespace characters inside a command key word are not allowed. For example, „SOUR ce“ is no valid command.
- Whitespace characters are required to separate the parameter from the prepending command. For example the command „SOURce:VOLTage40.5V“ will lead to a syntax error. Correctly you have to write „SOURce:VOLTage 40.5V“.
- Commas are separating different parameters of a sub system command.
- „Common Commands“ like *RST are not regarded as sub system commands. They are no part of a command path.

Units

- SCPI supports the usage of physical units. If no unit is indicated, SCPI assumes the basic unit. For example, the following commands are the same:

```
Volt 29.5V
sour:volt 29.5
Volt 29500mV
```

Numbers

ordinary numbers:	Numbers with sign, decimal point and exponent, when indicated, e.g.: `12.0, -5.34, 1000E-3` Physical units can be attached, e.g: `100mA, 50V`
MIN/MAX	`MINimum`: e.g. 0 Volt and 0 Amps `MAXimum`: rated voltage or rated current
DEF	`DEFault` matches the pre-setting of the device, e.g. after power-on or a *RST
boolean parameters	`ON` and `OFF` E.g. `OUTPut ON` The response to a query (`OUTP?`) is `0(OFF)` or `1(ON)`.

4.2 Command Sequence

All commands are executed in a sequential manner, in the same order they were arriving by the interface wire. However the complete execution of a command is not mandatory before starting the execution of the next command.

For example the command „*SOURce:VOLTage 100*“ prompts the power supply to adjust 100 volts at the output terminals. While the output capacitor of the power supply is charged and the voltage settles to the required 100 volts, the next command is already processed. The *SOURce:VOLTage*–command and the following command are executed quasi concurrently.

However this behaviour is not desirable at all times. In some cases the voltage, provided at the output terminals, has to be settled to the required value, before the next command shall be executed. Hence with the following **synchronization commands** you can enforce a strictly sequential processing.

Except for the *SOURCE*-subsystem all of the commands are processed sequentially in this power supply. Only the *SOURCE*-subsystem, used to adjust voltage and current, are processed concurrently. A sequential processing can be accomplished with the commands **OPC*, **OPC ?* and **WAI.*

4.2.1 *OPC – Operation Complete

After writing the command **OPC*, the device sets the bit “Operation Complete” (*OPC*, bit 0) of the Standard-Event-Status-Register to *TRUE*, as soon as all commands, which were received before, are completely processed. **OPC* should be used only as the last command in a terminated sequence of commands. The **OPC* –mechanism is resetted at power-on, when a Device-Clear-Message (*DCL*) or a **RST* occurs, or with a **CLS*.

Example: *SOURce:VOLTage 100;*OPC*
The *OPC*-bit will be set, as soon as the voltage
applies stable at the output terminals
(*or as soon as the programmed current limit is reached*).

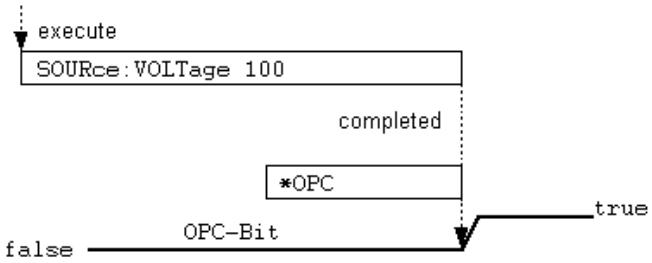


Fig. 4: Execution of the *OPC-command

4.2.2 *OPC? – Operation Complete Query

After receiving the command **OPC?*, the device writes the ASCII character „*1*“ (31hex) in the output buffer, as soon as all commands, which were received before, are completely processed. **OPC?* should be used only as the last command in a terminated sequence of commands.

Example: *SOURce:VOLTage 100; *OPC?*

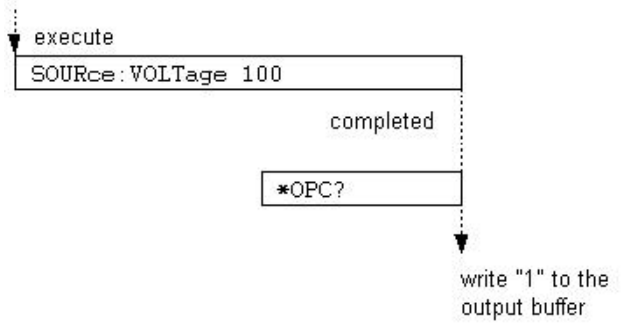


Fig. 5: Execution of the *OPC?-command

4.2.3 *WAI – Wait-to-Complete

After receiving the command **WAI*, the device waits with execution of following commands until all preceding commands are processed.

Example: *SOUR:VOLT 100; *WAI; :MEAS:CURR?*
Provides 100 volts and waits until the voltage has settled
(*or as soon as the programmed current limit is reached*).
Then the output current is sensed.

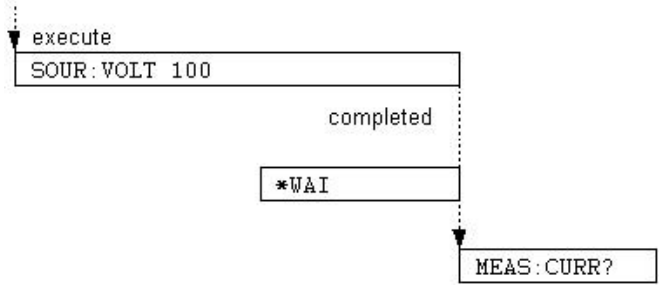


Fig. 6: Execution of *WAI-command

4.3 Status System

The status system enables the application programmer to retrieve informations about the current condition of the instrument in order to react accordingly. If, for example, an error or any other event occurred during the processing of the application, the status system has recorded the occurence of the error or event.

The programmer is free to adjust which conditions and events are to be recorded by setting mask bits of the appropriate enable registers.

The status system is a strict hierarchical construction. The status byte represents the top level. It is composed of the sum-bits of subordinate status registers (see Fig. 7).

Service Request

Instruments with IEEE488 interface are able to signal their device state to the application software by pulling the so called service request line (SRQ) to LOW. By means of the SRQ the system controller, usually a personal computer equipped with IEEE488 interface controller board, realizes if a device requests service. Subsequently the controller polls the bus devices one after the other using the “serial poll” method. The devices themselves are responding to a serial poll with their status byte. The device, which has caused the SRQ by pulling the service request line, is identified, because bit 6 of the status byte is set TRUE. Bit 6 is called RQS bit, request service bit.

By means of masking the bits of the status request enable register (SRE) the programmer can decide, which event shall trigger a service request. With command **SRE* („Service Request Enable“) the SRE-register is written.

For example command *`*SRE 16`* sets bit 4 of the service request enable register. This bit enables a SRQ caused by the so called MAV bit (“message available”). The MAV bit is set, as soon as data are available in the output buffer of the device.

After power-on the service request enable register is cleared. Mask bits, which should enable a SRQ, have to be set explicitly to “1” by the application software.

The status byte in conjunction with the service request is the only way of a instrument to interrupt a running application and draw the attention of the bus controller.

4.3.1 Diagram of the status system

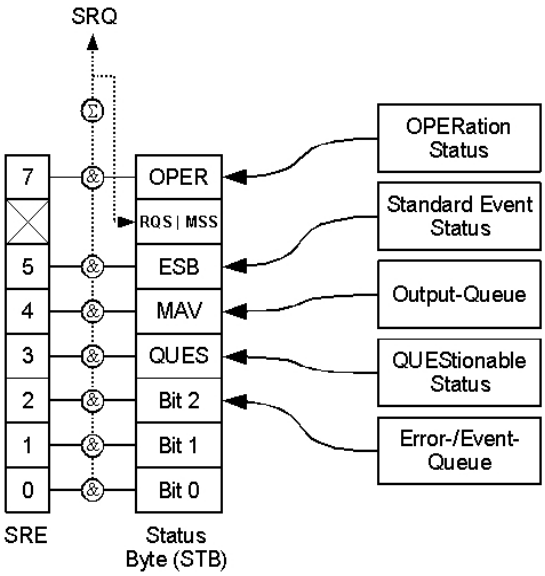


Fig. 7: Status Byte

Figure 7 shows the elements of the status system. ⊛ matches a logical AND, Σ a logical OR.

Subordinated to the status byte are the other status registers:

- OPERation status
- Standard Event Status Register (ESR)
- QUEStionable status

and the queues

- output queue (output buffer)
- Error/Event queue

The subordinated status registers and queues are mapped onto the appropriate sum bits of the status byte. For example, if there are data available in the output queue, then bit MAV of the status byte is set. If there is an entry in the error/event queue, then bit 2 of the status byte is set.

You have to set the dedicated bit of the service enable register (SRE), if you want to raise a service request (SRQ).

Table 7 shows the meaning of the bits of the status byte.

status byte:			
bit-nr.	abbr.	name	meaning
7	OPER	OPERation Status	sum bit of the Operation Status structure

status byte:			
6	RQS	Requested Service	Notifies, that a device requests service and a SRQ interrupt was raised. The RQS bit is the response to a serial poll. The controller realizes, that the SRQ was triggered by the device with the RQS bit set to “1”.
	MSS	Master Summary Status	The MSS bit is provided in response to a **STB?*-query. The MSS bit is cleared, if the subordinated status registers are cleared or if the SRE register is cleared with **SRE 0*.
5	ESB	Event Summary Bit	sum bit of the Event Status Registers
4	MAV	Message Available	The output buffer contains data, which can be read by the application.
3	QUES	QUEStionable Status	sum bit of the QUEStionable Status structure
2	Error/Event Queue		Error/Event queue is not empty

Table 7: Status Byte

Commands regarding the status byte:

**STB? – status byte query*

Returns the data content of the status byte. Bit number 6 is assigned to the MSS bit. The status byte is not changed or cleared by reading with *STB?. The data content only changes, if the subordinated status registers, which form the status byte with their sum bits, are cleared.

**SRE <0..255> – Service Request Enable - command*

Sets the SRE register to the desired value. The setting of a bit in this register enables a SRQ interrupt, as soon as an according event occurs.

**SRE? – Service Request Enable - query*

Reports the content of the SRE register. Bit 6 is always reported with 0.

Examples:

**STB?* Reports decimal 20, for example, corresponding to binary 00010100_b. That is, bit 2 (error-/event queue) and bit 4 (MAV) are set TRUE.

**SRE 12* 12 decimal corresponds to binary 00001100_b. The device triggers a SRQ, if there is an item in the error-/evtent queue (bit 2) or if the sum bit QUES was set (bit 3).

4.3.2 Subordinated status registers and queues

Like already shown in Fig. 7, the status system comprises the following subordinated status registers:

- Standard Event Status Register (ESR)
- OPERation Status
- QUEStionable Status

Each of these status registers forms a sum bit, which is projected onto the status byte. In this way it is possible

to trigger a SRQ caused by the subordinated status registers.

Additionaly the condition of the instrument is indicated by the

- output buffer
- error-/event queue

If there is an entry in one of the queues, then the according bit in the status byte will be set.

4.3.2.1 Standard Event Status Register

The standard event **status** register is represented by the sum bit ESB. The activation of the sum bit can be masked using the standard event **enable** register (see Fig. 8).

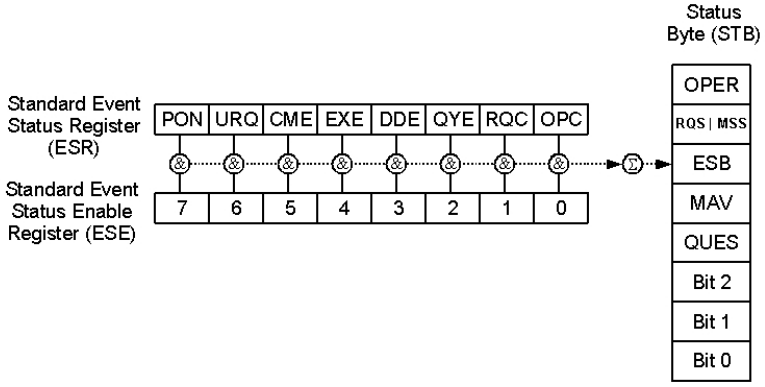


Figure 8: Standard Event Status Register

Standard Event Status Register:			
bit-nr.	abbr.	name	meaning
7	PON	power ON	This bit is set, when power is switched on
6	URQ	user request	- not used -
5	CME	command error	This bit is set, if a syntax error occurs. An error is placed into the error-/event queue with error number -100 to -199 (see error codes, chapter 6)
4	EXE	execution error	This bit is set, if a command can not be executed, because the range of values is exceeded, for example. An error is placed into the error-/event queue with error number -200 to -299 (see error codes, chapter 6)
3	DDE	device dependent error	This bit is set, if a device dependent error occurs. An error is placed into the error-/event queue with error number -300 to -399 (see error codes, chapter 6)
2	QYE	Query Error	This bit is set, if , for example, a command is sent, without having read the response of the preceding command. An error is placed into the error-/event queue with error number -400 to -499 (see error codes, chapter 6)
1	RQC	Request Control	- not used -
0	OPC	Operation Complete	This bit ist set after a `*OPC` command, if all previous commands are executed (see also 4.2 – command sequence)

Table 8: meaning of the bits comprised by the standard event status register

Commands regarding the standard event status register:

**ESR? - Standard Event Status Register - query*

Reports the content of the standard event status register (ESR). **The register is cleared when queried.**

**ESE <0..255> – Standard Event Status Enable*

Sets the ESE register to the desired value. The setting of a bit in this register enables the according event to set the sum bit ESB to TRUE within the status byte.

**ESE? – Standard Event Status Enable - query*

Returns the data content of the ESE register. The content of the register is left unchanged when queried with *ESE?

Examples:

**ESR?*	Reports decimal 21, for example, corresponding with binary 00010101_b. That is, bit 0 (OPC), bit 2 (QYE) and bit 4 (EXE) are set TRUE within the ESR register. **ESR is cleared after queried with *ESR?**
*ESE 24	Decimal 24 corresponds with binary 00011000b. Thus the sum bit ESB is set, after a device dependent error (bit 3) or an execution error (bit 4) occur.

4.3.2.2 QUEStionable Status

Fig. 9 shows the configuration of the QUEStionable status register. The meaning of the individual bits is explained in Table 9.

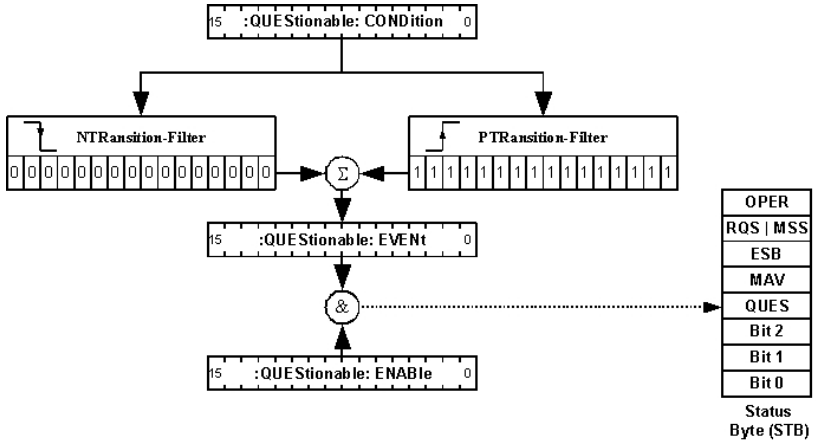


Figure 9: QUEStionable Status Register

STATus:QUEStionable:CONDition - Register			
bit-Nr.	abbr.	name	meaning
11 ... 15			- not used -
10	WDTR	Watchdog-Timer-Reset	A device reset was caused by a watchdog overflow.
9	OVP	Over Voltage Protection	This bit shows that an overvoltage event tripped the over voltage protection unit.
5 ... 8			- not used -

STATus:QUEStionable:CONDition - Register			
4	TEMP	over temperature	This bit is set, if an excessive temperature is sensed inside of the device.
2,3			- not used -
1	CURR	CURRent	This bit is set, if the measured electrical current differs from the desired current level (programmed current limit).
0	VOLT	VOLTage	This bit is set, if the measured voltage differs from the desired voltage level.

Table 9: bits of the STATus:QUEStionable Register

Between CONDition and EVENt register has to be distinguished. The CONDition register reports a device condition, defined by the individual bits. The VOLT bit, for example, indicates, that the output of the power supply has not yet settled to the programmed voltage. If the programmed value is reached, then the VOLT bit is set to FALSE.

The EVENt register, on the contrary, signal the change of a bit inside the CONDition. For example, if an over voltage condition occurs, then bit OVP of the CONDition register alters from “0” to “1”. The event is recorded with the EVENt register.

Intention of the EVENt register is to notifiy the user that an event has occured. The EVENt register is cleared, as soon as it is read with the query `STATus:QUEStionable[:EVENt]?`. In contrast the CONDition register remains unchanged, because it is solely dependent on the device conditions, not on queries polled by the user.

Transition filters according to Figure 9 are integral part of the SCPI status system. They are working like edge detectors and are fed by the CONDition register. The change of a bit within the CONDition register is realized by the transition filter and passed on to the EVENt register, where it can be checked by the user.

The transition filters are not programmable. All of the bits in the PTRansition filter are fixed to “1”, all bits of the NTRansition filter are fixed to “0”. That is, only events are recorded in the EVENt register, which change from “0” to “1” inside the CONDition register, but not from “1” to “0”. The recorded bits remain inside the EVENt register, as long the EVENt register is read by the user.

Examples:

`STATus:QUEStionable?`	Returns the content of the QUEStionable:EVENt register.

`STAT:QUES:ENAB 528`
or
`STAT:QUES:ENAB #H210`
	528 in decimal notation corresponds to the binary number 0000 0010 0001 0000b. If bit 4 (temperature high) or bit 9 (over voltage protection) appear in the STAT:QUES:EVENt register, then sum bit QUES is set to “1” within the status byte.

`STAT:QUES:CONDition?`	Queries the QUEStionable-CONDition register.

4.3.2.3 OPERation Status

The OPERation status register are arranged just as the QUEStionable status registers. However the power supply makes no use of the `STATus:OPERation:CONDition` register.

It is implemented because of compliance the SCPI standard.
Readings with `STATus:OPERation:CONDition?` return always 0.

4.3.2.4 Output-Queue

The device keeps a reply within the output buffer until the buffer is read by the GPIB controller. Available data in the output buffer are indicated by the bit MAV (message available), placed as bit 4 in the status byte.

If the device is addressed as talker and the output buffer holds no data, then the error „QUERY UNTERMINATED“ is generated. The device sends no data and the GPIB controller will run into a timeout condition.

If the device recieves a message, although still holding data within the output buffer, then the error „QUERY INTERRUPTED“ is generated. The output buffer is cleared and the received message will be processed.

4.3.2.5 Error/Event-Queue

Errors and events, which occur inside the device, are recorded by the status and EVENt registers by setting the according bits, see above. The bit values are recorded, until the EVENt register is read by the controller or the command *`*CLS`* (clear status) is received.

However, the order, in which the errors and events occurred, can not be determined by evaluating the status registers. For this purpose the error/event queue stores the occurence of errors and events sequentially. The error-/event queue relates to a FIFO buffer (first in, first out). This means, if the queue is queried with *`SYSTem:ERRor?`* the error will be reported, which occurred at first. Then the second one, the third one and so on.

Each call of the command *`SYSTem:ERRor?`* delivers one entry from the error-/event queue. The command can be repeated until there are no more entries left in the queue. Then the device responds with a *`"0, No error"`*.

If the error-/event queue contains any error, then bit 2 of the status byte is set concurrently. This bit can be used to create a SRQ on error conditions.

4.3.3 Parallel Poll

Rarely used is the so called parallel polling. In contrast to the serial poll the parallel poll queries the state of several devices at the same time. A device responds with the “ist”-flag (individual status flag) to a parallel poll. Because the status information is made up of one bit and there are 8 single data lines, hence eight individual devices can be polled concurrently.

Each device, which shall attend the parallel poll, has to be configured before. The device is notified of the data line (1 to 8), where the ist-flag has to be put on. Moreover it has to be configured if a 1 or a 0 shall be put on the data line, when „ist=TRUE“.

Figure 10 shows in which way the ist flag is generated. It is the output of the the logical AND between status byte (STB) and the PRE mask register (PRE = parallel poll request enable).

The PRE register can be written with the command *`*PRE <0..255>`* . In addition to the parallel poll the PRE register can also be read by the command *`*IST?`*.

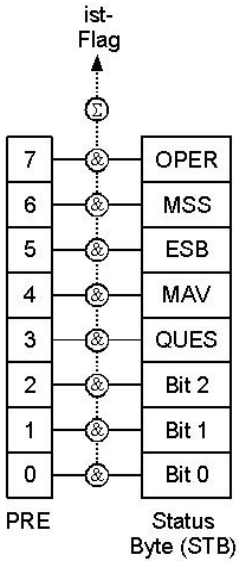


Fig. 10: forming the ist flag

5 SCPI commands

This chapter describes all commands which are implemented in the present power supply. The notation complies with SCPI standard version 1999.0.

Each command and each command subsystem respectively is introduced in a short table showing the appropriate parameters. The commands are described in detail afterwards.

Long and short form of commands

Usually in manuals the short form of a command or of a key word is written in upper case. The device itself does not distinguish between upper and lower case.

Command or query

Generally for each command an appropriate query exists. The query is expressed simply by adding a “?” to the command.

If a command does not exhibit its appropriate query or if a command can be written exclusively in query form, this case is notated as a comment in the right column of the table.

5.1 Common Commands

Common commands are available in all SCPI devices. They are essential already in standard IEEE 488.2, in the same manner in SCPI. Many common commands are related to the status system, which is described in chapter 4.

Command	Parameter	Comment
***CLS**		Clear Status no query
***ESE**	0 ... 255	Event Status Enable
***ESR?**		Standard Event Status Query only query
***IDN?**		Identification Query only query
***IST?**		Individual Status Query only query
***OPC**		Operation Complete
***PRE**	0 ... 255	Parallel Poll Register Enable
***RST**		Reset no query
***SRE**	0 ... 255	Service Request Enable
***STB?**		Read Status Byte Query only query
***TST?**		Self Test Query only query
***WAI**		Wait-to-Continue no query

5.1.1 *CLS

Clear Status deletes all event registers and the error-/event queue. That is, all events are removed from the status system, which are not queried by the application yet. The settings of the status system, represented by the mask registers, are not affected.
These registers are cleared:

- Standard Event Status Register (ESR)
- EVENt-part of the QUEStionable-Register
- EVENt-part of the OPERation-Register
- Error/Event-Queue

5.1.2 *ESE

Standard Event Status Enable sets the Standard Event Status Enable register to the stated value. Bit 5 of the status byte (ESB = Event Summary Bit) is formed out of the AND relation of ESR and ESE registers. The query *ESE? returns the content of the ESE register in decimal form.

5.1.3 *ESR?

Standard Event Status Query returns the content of the Standard Event Status register in decimal form. ESR register and ESB bit within the status byte are cleared.

5.1.4 *IDN?

Identification Query returns a device identification as a character string. For example the answer is

„TET, ARGOS 1000, 12345, 1.03“

TET:	manufacturer
ARGOS 1000:	device model
12345:	serial number
1.03:	firmware revision number

5.1.5 *IST?

Individual Status Query returns the content of the ist flag (0 or 1). The ist flag matches the status bit, which is transmitted on a defined data line after a parallel poll.

5.1.6 *OPC

Operation Complete sets bit 0 of the Standard Event Status register, if all preceding commands have been processed. This bit can be used to trigger a service request. *`*OPC`* should be used as last command in a terminated command sequence.

Example: `SOURce:VOLTage 100V;*OPC`

The *OPC* bit is set inside of the Standard Event Status register,
as soon as the output voltage has settled to a value within the tolerances of the output terminals (or if the current limitation is activated).

5.1.7 *OPC?

Operation Complete Query writes a „1“ to the output buffer, as soon as all preceding commands have been processed. *`*OPC?`* should be used as last command in a terminated command sequence.

5.1.8 *PRE

Parallel Poll Register Enable writes the stated value to the Parallel Poll Enable register. This register determines, which bits of the status byte are summed to the ist flag. The ist flag is transferred during a parallel poll.

5.1.9 *RST

Reset puts the device into a defined basic state. For each function this basic state is notated as *RST value. *RST does not change any event- or enable registers of the status system. The output buffer is not cleared by the *RST command. Voltage and current limit are put down to 0, the overvoltage protection is set to 120% of the nominal voltage. The output is switched on (OUTPut = ON).

5.1.10 *SRE

Service Request Enable writes the stated value to the Service Request Enable register. The SRE register configures which bits of the status byte are allowed to trigger a service request. The query *SRE? returns the content of the Service Request Enable register in decimal form. Bit 6 is read always with 0.

5.1.11 *STB?

Status Byte Query reads the content of the status byte in decimal form. Bit 6 is called MSS. The MSS bit shows, that a device has requested service, even if the device was already serial polled and the RQS bit was cleared after polling.

5.1.12 *TST?

Self Test Query executes an internal self test of the device. It responds with the appropriate error code in decimal form, “No Error”.

5.1.13 *WAI

Wait-to-Continue permits the execution of subsequent commands not until preceding commands are processed completely.

5.2 Special RS232 commands

When using GPIB the device enters remote mode automatically when the first message is received. If required the device can be brought back to local mode using the IEEE 488.1 command <GTL> (go to local).

If the power supply is controlled via RS232, unlike GPIB the change-over from local to remote mode and vice versa has to be done with explicit control commands.

5.2.1 @REM

With @REM the power supply changes to remote mode. Remote mode is indicated by the front panel LED „REM“.

5.2.2 @LOC

With command @LOC the power supply switches to local mode.

Command	Parameter	Comment
@REM		REMOTE change-over only RS232 no query
@LOC		LOCAL change-over only RS232 no query

5.3 MEASure – Subsystem

Command	Parameter	Comment
MEASure **[:SCALar]** **:VOLTage[:DC]?** **:CURRent[:DC]?** **:AUXiliary[:DC]?**		 only query only query only query

5.3.1 MEASure[:SCALar]:VOLTage[:DC]?

With this command the output voltage of the power supply is sensed. The data format is floating point. The unit equals Volts.

example: `'MEAS:VOLT?'`
response e.g.: `'23.507'`

please note: `'MEAS:VOLT?'` and `'VOLT?'` are different commands.
While `'MEAS:VOLT?'` senses the actual voltage provided at the output terminals, in contrast `'VOLT?'` shows the last transmitted setpoint value.

characteristics: *RST-value: --
SCPI: conform

5.3.2 MEASure[:SCALar]:CURRent[:DC]?

With this command the output current of the power supply is sensed. The data format is floating point. The unit equals Amps.

example:	`'MEAS:CURR?'` response e.g.: '56.466'	
please note:	`'MEAS:CURR?'` and `'CURR?'` are different commands.	
	While `'MEAS:CURR?'` senses the actual current provided at the output terminals,	
	in contrast `'CURR?'` shows the last transmitted setpoint value.	
characteristics:	*RST-value:	--
	SCPI	conform

5.3.3 MEASure[:SCALar]:AUXiliary[:DC]?

This command reads the actual setting of the overvoltage protection. The data format is floating point. The unit equals Volts.

example:	`'MEAS:AUX?'` response e.g.: `'60.229'`	
characteristics:	*RST-value:	--
	SCPI:	device specific

5.4 OUTPut – Subsystem

Command	Parameter	Comment
OUTPut **[:STATe]**	<Boolean>	

5.4.1 OUTPut[:STATe] ON|OFF

With this command the output is switched on or off.

example:	`'OUTP OFF'`	
	`'OUTP?'`	response e.g.: `'0'`
characteristics:	*RST-value:	ON
	SCPI:	conform

5.5 SOURce – Subsystem

Command	Parameter	Comment
[SOURce]		
:CURRent		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:VOLTage		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:PROTection		
:CLEar		no query
[:LEVel]	<numeric_value>	
:TRIPped?		only query

5.5.1 [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] <numeric_value | MIN | MAX>

This command adjusts the set point of the output current and the current limit respectively.

examples:	'CURR 2.34'	set point = 2.34A
	'CURR min'	set point = 0A
	'CURR max'	set point = nominal current
	'CURR 1500mA'	set point = 1.5A
	'CURR?'	response: '1.5'
characteristics:	*RST-value:	0 Ampere
	SCPI:	conform

5.5.2 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric_value | MIN | MAX>

This command adjusts the set point of the output voltage.

examples:	'VOLT 12.0V'	set point = 12V
	'VOLT MIN'	set point = 0V
	'VOLT MAX'	set point = nominal voltage
	'VOLT 100'	set point = 100V
	'VOLT?'	response: '100'
characteristics:	*RST-value:	0 Volt
	SCPI:	conform

5.5.3 [SOURce:]VOLTage:PROTection:CLEar

This command resets a tripped overvoltage protection (OVP).

example:	'VOLT:PROT:CLE'	
characteristics:	*RST-Wert:	--
	SCPI:	conform

5.5.4 [SOURce:]VOLTage:PROTection[:LEVel] <numeric_value | MIN | MAX | DEF>

This command adjusts the point, at which the overvoltage protection (OVP) trips the output voltage. After switching on the device or after a *RST it is set to 120% of nominal voltage.

examples:	'VOLT:PROT 48V'	OVP = 48 Volts
	'VOLT:PROT min'	OVP = 0 Volts
	'VOLT:PROT max'	OVP = 120% of nominal voltage
	'VOLT:PROT DEF'	OVP = 120% of nominal voltage
	'VOLT:PROT?'	response: '48'
characteristics:	*RST- value:	120 % of nominal voltage
	SCPI:	conform

5.5.5 [SOURce:]VOLTage:PROTection:TRIPped?

This command reads the state of the overvoltage protection (OVP). If OVP has tripped, it returns a “1” else a “0”.

example: 'VOLT:PROT:TRIP?' response: '0' or '1'
characteristics: *RST- value: --
SCPI: conform

5.6 STATus – Subsystem

The status subsystem is based on the commands used for masking and reading the OPERation- and QUEStionable registers (see chapter 4.3). The status system is not affected bei the reset command *RST. Instead of *RST the clear status command (*CLS) can be used to reset OPERation and QUEStionable status. *CLS clears all event registers and the error-/event queue.

Command	Parameter	Comment
STATus		
:OPERation		
:CONDition?		only query
:ENABle	0 ... 65535	
[:EVENt?]		only query
:PRESet		no query
:QUEStionable		
:CONDition?		only query
:ENABle	0 ... 65535	
[:EVENt?]		only query

5.6.1 STATus:OPERation:CONDition?

Reads the content of the STATus:OPERation:CONDition register. The register remains unchanged. It depends on the hardware state of the device. The register can be read by a query, but it can not be changed. The meaning of the individual bits of the CONDition register is described in chapter 4.3. A query reports a decimal number in the range of 0 ... 65535, which matches the content of a 16 bit register.

example: 'STAT:OPER:COND?' response '0'
characteristics: *RST- value: --
SCPI: conform

5.6.2 STATus:OPERation:ENABle 0 ... 65535

This command writes the bits of the ENABle part of the STATus:OPERation register. Occuring events which are reported within the STATus:OPERation part are setting the sum bit „OPER“ (bit 7 of the status byte), assumed the according events are enabled. See chapter 4.3 for details.

example: 'STAT:OPER:ENAB #H3039'
'STAT:OPER:ENAB?' response '12345'

characteristics: *RST- value: --
SCPI: conform

5.6.3 STATus:OPERation[:EVENt]?

This command reads the content of the STATus:OPERation[:EVENt] register. A query reports a decimal number in the range of 0 ... 65535, which matches the content of the 16 bit register. The EVENt register is cleared when it is queried.

example: 'STAT:OPER?'
characteristics: *RST- value: --
SCPI: conform

5.6.4 STATus:PRESet

This command resets the ENABle parts of the STATus:OPERation and STATus:QUEStionable registers. The registers Standard Event Enable (ESE) and Service Request Enable (SRE) remain unchanged. They have to be cleared explicitly with the commands *'*ESE 0'* respectively *'*SRE 0'.*

example: 'STAT:PRES'
characteristics: *RST- value: --
SCPI: conform

5.6.5 STATus:QUEStionable:CONDition?

Reads the content of the STATus:QUEStionable:CONDition register. The register remains unchanged. It depends on the hardware state of the device. The register can be read by a query, but it can not be changed. The meaning of the individual bits of the CONDition register is described in chapter 4.3. A query reports a decimal number in the range of 0 ... 65535, which matches the content of a 16 bit register.

example: 'STAT:QUES:COND?'
characteristics: *RST- value: --
SCPI: conform

5.6.6 STATus:QUEStionable:ENABle 0 ... 65535

This command writes the bits of the ENABle part of the STATus:QUEStionable register. Occuring events which are reported within the STATus:QUEStionable part are setting the sum bit „QUES“ (bit 3 of the status byte), assumed the according events are enabled. See chapter 4.3 for details.

example: 'STAT:QUES:ENAB 12345'
'STAT:QUES:ENAB?'
characteristics: *RST- value: --
SCPI: conform

5.6.7 STATus:QUEStionable[:EVENt]?

This command reads the content of the STATus:QUEStionable[:EVENt] register. A query reports a decimal number in the range of 0 ... 65535, which matches the content of the 16 bit register. The EVENt register is cleared when it is queried.

example: 'STAT:QUES?'
characteristics: *RST- value: --
SCPI: conform

5.7 SYSTem – Subsystem

Command	Parameter	Comment
SYSTem		
:ERRor		
[:NEXT]?		only query
:VERSion?		only query

5.7.1 SYSTem:ERRor[:NEXT]?

This command reads the oldest entry of the error queue and thereby clears this entry. Positive error numbers indicate device specific errors, negative numbers are reserved to SCPI errors. If the error queue is empty, a '*0, "No error"*' is returned, otherwise the corresponding error code.

SYSTem:ERRor:NEXT? can be repeated as long as all collected errors are reported and finally a '*0, "No error"*' is returned. Possible error codes are listed in chapter 6.

example: 'SYST:ERR?'
characteristics: *RST-value: --
SCPI: conform

5.7.2 SYSTem:VERSion?

This command reads the version of the SCPI standard, which the device complies to. The response is in the format YYYY.V, for example 1990.0.

example: `'SYST:VERS?'`
characteristics: *RST-value: --
SCPI: conform

6 Error and Event numbers

6.1 SCPI error and event numbers

Code	*error text*	*comment*
0	**No error**	error queue contains no entries
-100	**Command Error**	incorrect command
-102	**Syntax Error**	incorrect syntax
-108	**Parameter not allowed**	Incorrect parameter or too many parameters
-109	**Missing Parameter**	One or more parameters are missing
-110	**Command Header Error**	Command header is incorrect.
-115	**Unexpected Number of Parameters**	
-200	**Execution Error**	An error occured during execution of a command
-220	**Parameter Error**	Incorrect parameter(s)
-222	**Data out of range**	A value is out of range.
-240	**Hardware Error**	Command can not be executed because of missing hardware
-300	**Device-specific error**	
-350	**Queue Overflow**	This error occurs, if the error queue is full of entries. The error queue can contain a maximum of 5 entries.
-400	**Query Error**	
-410	**Query Interrupted**	A query was interrupted, e.g. the device gets new commands, before remaining data within the output queue are sent out.
-420	**Query Unterminated**	A query was not terminated with the expected message termination.
-430	**Query Deadlocked**	A query can not be executed, because input and output buffer are full of data.

Code	*error text*	*comment*
-500	**Power On**	
-800	**Operation Complete**	All commands have been executed (see *OPC and *OPC?)

6.2 Device specific error messages

Code	*error text*	*comment*
1xx	**internal Hardware Error**	
200	**UART Error**	incorrect baudrate, parity ...
210	**DAC Error**	D/A-converter failed
220	**Flash Write Error**	Error, when saving a parameter
230	**Flash Erase Error**	Error, when erasing internal flash memory
260	**Temperature Error**	Temperature exceeds limit
270	**Overvoltage Protection Tripped**	A voltage greater than OVP occured.
280	**Watchdog Timer**	Watchdog Timer has caused a processor reset

7 TET specific commands

7.1 Hints for Programming

Attention should be paid to:

1. If TET specific commands shall be used, you have to set jumper J4 “SYNTAX” on the interface card. See Fig. 1. It is not allowed to mix TET commands and SCPI commands.
2. The device does not conform to the IEEE 488.2 standard, if the proprietary TET commands are use. Particularly the status byte and the treatment of the service request (SRQ) differ from the standard (see chapter 8.4).
   „Common Commands“, described in chapter 5, are not available, except of the commands *`*IDN?`* and *`*STB?.`*
3. Each input of setpoint values take effect only with the enabling character X, e.g. *`V30X`*.
4. The overvoltage protection has to exceed the desired output voltage with more than 10% or more than 1 volt. E.g. if an output voltage of 40 volts is desired, the OVP has to be adjusted to a minimum of 41 volts. By default OVP is set to 120% of the rated voltage.
5. The device expects commands in ASCII code, however upper and lower case letters are accepted. The data output is made up of upper case letters.
6. When transmitting data, each data block is attached with a termination characater (e.g. *`EOI`*) by the talker. To enable a communication between computer and power supply, the interface card of the power supply as well as the interface card of the controlling computer have to be set to the same termination character (see chapter 2).
7. In contrast to SCPI- respectively the IEEE 488.2 – standard, where queries are replied but commands are not, the proprietary TET commnand set replies all messages, no matter if query or simple command (see command list).
8. Concatenation of commands is possible by separating the individual commands by a semicolon „;“. The responses to the individual commands are separated by a comma within the reply string. Example:
   Command: *`f0; l120x; f1`* Response: *`F_0, >, F_1`*

7.2 Programming of Setpoints

7.2.1 Input Mode

The setpoints of voltage (V), current (C) and overvoltage protection (L) can be entered as percentage of the nominal values or using the floating point format.

By default the power supply works with the floating point format.

The percentage mode is enabled with the command „*`F0`*“.

„*`F0`*“: Percentage mode

„*`F1`*“: Floating point mode

Please note that all setpoint inputs take effect not before the enabling command „X“.

With the command „N“ the output voltage can be set to 0 volts without changing the setpoint value. „N“ disables the adjusted setpoints.

Entering the character „*`X`*“ cancels the command „*`N`*“ and the output voltage settles to the last adjusted output voltage. „X“ enables the adjusted setpoints.

7.2.2 ExamplesBeispiele

V0C0L0X	- set voltage, current and overvoltage protection to 0
F1 *V10.25X* *C100X* *L1.2E2X*	- change to float mode - adjust voltage to 10,25 Volts - adjust current limit to 100 Amps - adjust overvoltage protection to 120 Volts
F0 *V75.23X* *L120X* *N* *X*	- change to percentage mode - set voltage to 75,23% of nominal voltage - set overvoltage protection to 120% of nominal voltage - disable setpoints (voltage is set to 0 volts) - enable setpoints (voltage adjusts to the previous setpoint)

Table 10: Examples how to program setpoints

7.3 Measuring the actual voltage and current

With the commands M0 to M7 the desired measuring mode can be selected. The measuring mode determines, which actual value (voltage, current or AUX) will be transmitted by the power supply when the command „M“ has been received:

Measuring mode	***Actual Value to be transmitted***
M0	no value
M1	voltage
M2	current
M3	voltage and current (default after switching on the power supply)
M4	AUX (Atlas, Argos, Hellas, Hercules = OV-limit)
M5	voltage and AUX
M6	current and AUX
M7	voltage, current and AUX
Mess	The actual values, which were configured by M0 to M7, are dumped continuously to the serial port and can be viewed e.g. by a terminal software.

Tablee 11: Measuring modes

Caution: Actual values are always transmitted in floating point notation, even if the percentage mode was adjusted with „*F0*“.

Example:

Command	***Response***
M3	*>*
M	*01_V:23.12340000_C:2.434500000*

Command	*Response*
M7	*>*
M	*01_V:23.12340000_C:2.434500000_X:48.0230000*

Table 12: Measuring modes M3 and M7

7.4 Status Byte

The reporting of the device status does not conform with IEEE 488.2 if the proprietary TET commands are used. However in order to support control software, which was written specific to the older versions of the Option 34, an adaptation to IEEE 488.2 was abandoned. If a control software needs status reporting according to IEEE 488.2, it should be written with SCPI syntax instead of using proprietary commands.

Monitoring of the status byte can be used to detect an activation of the current limitation (CC) and/or the overvoltage protection.

7.4.1 Configuration of the Status Byte

Bit	*Name*	*Meaning*
0	OVM	Present condition of the overvoltage protection
1	CCM	Present condition of the current limit
2	OVG	Latched condition of the overvoltage protection
3	CCG	Latched condition of the current limit
4	0	
5	0	
6	SRQ	SRQ request
7	ERR	error has occured -> see error queue

7.4.2 Reading the Status Byte

with command „W“:

After sending the letter „*W*“ the interface card of the power supply returns the status byte in the form of „*01_W_xxxxxxxx*“.

If the current limit of the device is active at the moment of reading, the command „*W*“ returns „*01_W_00001010*“.

with a serial poll:

The power supply has to be configured with the command “Q” in such a way, that a serial request (SRQ) will be generated, if an overvoltage protection (OVP) or a current limitation (CC) have occured. If the SRQ line has been triggered, the GPIB controller polls all of the devices connected to the GPIB bus. Each device answers with its status byte to a serial poll. Bit 6 of the status byte defines, if the device has requested service.

7.4.3 Resetting the status byte

The generation of a service request (SRQ) can be disabled with DIL switch S1.7 (ON = SRQ enabled, OFF = SRQ disabled).

The status byte can be resetted with the commands *&0* respectively *&1*. Entries within the error queue are cleared in doing so.

&0: reset status byte and disable further SRQs.
&1: reset status byte and enable further SRQs.

The commands *&0* and *&1* overwrite the settings of DIL switch S1.7. E.g. if the device was configured with S1.7 to disable SRQs, then this setting can be changed with *&1*.

7.4.4 Masking of the SRQ generation

With the command „Q“ the generation of a SRQ can be masked to the occurence of the overvoltage protection (OV) and/or the current limitation (CC) (see table 13).

command	***masking***
Q00	no SRQ
Q01	SRQ if CC
Q10	SRQ if OV
Q11	SRQ if CC or OV

Table 13: Masking of SRQ

7.5 Error Handling

Occuring errors are buffered in the error queue which acts as a FIFO buffer. The queue contains the last 5 errors.

If the queue already contains 5 errors and a further error occurs, then the last error number is replaced with number -350, Queue Overflow.

Command “Y” requests the oldest entry of the error queue. The entry is cleared when requested.

Positive error numbers identify device specific errors, negative error numbers mark dedicated SCPI error messages. If the error queue is empty, then a 0, „No error“ is returned, otherwise the corresponding error number.

The command „Y“ can be executed as long as errors are existent within the queue and at last a „0“, No Error is returned.

Potential error number are described in chapter 6.

7.6 Command List

Each command is acknowledged by the power supply, either with the requested data or with a simple character

„>“ : Acknowledge, i.e. command has been executed

„!“ : Error has occured, command has not been executed

***IDN?**	Identification Query
Type:	Query
Use:	Identification of the device connected to the IEEE 488 bus
Query syntax:	*IDN?
Response:	String with following configuration: MANUFACTURER, MODEL, SERIAL NUMBER, FIRMWARE REVISION NUMBER

Example: TET, ARGOS 2000, 0, 1.0

#	Identification Query – identical with *IDN?
Type:	Query
Use:	Identification of the device connected to the IEEE 488 bus
Query syntax:	#

B	Remote Control (only RS232)
Type:	Command or Query
Use:	Switch the power supply to remote control
Command syntax:	B1: Remote Control B0: Manual Control
Query syntax:	B
Response:	B1: Remote Control B0: Manual Control
Default:	B0
Note:	If the power supply shall be controlled via RS232, the device has to be switched to remote control with command „B1“. Using IEEE 488 this is done automatically.

C	Current (adjust setpoint of current respectively current limit)
Type:	Command
Use:	Programming of the output current
Command syntax:	C<number>
Response:	> or !
Default:	C0
Note:	This command adjusts the current setpoint to the desired value. **Caution:** Take account of percentage/float mode.
Example: C50X	float mode: Switch current limit to 50 Amps percentage mode: Switch current limit to 50% of rated current

---

F	Float- / Percentage mode
Type:	Command or Query
Use:	Toggle between float and percentage mode
Command syntax:	F0: percentage mode F1: float mode
Query syntax:	F
Response:	F_0: percentage mode F_1: float mode
Default:	F1
Note:	After a switch-on the device expects the input of reference values of V, C and L in float mode. If required the user can change to percentage mode with command „F0“.
Example: F0 L120X F1	Switch to percentage mode. Set overvoltage protection to 120% of nominal voltage. Switch back to float mode.

I	Read digital inputs
Type:	Query
Use:	Reads the digital inputs DIN P1, DIN P2, DIN P3
Query syntax:	I
Response:	I_xyz with: x = DIN P1 : overvoltage protection (OV) y = DIN P2 : current limit (CC) z = DIN P3 : overtemperature (OT)
Default:	--
Example: I	Response: I_010

L	LIMIT (OV, Overvoltage Protection)
Type:	Command
Use:	Programming the Overvoltage Protection
Command sysntax:	L<number>
Response:	> or !
Default:	L = 120% Unom
Description:	This command programs the overvoltage protection to the desired value. **Caution:** Consider percentage/float mode !
Example: L30X	float mode: Overvoltage protection is set to 30 volts. percentage mode: Overvoltage protection is set to 30% of nominal voltage.

M	Metering
Type:	Command or Query
Use:	Metering of voltage, current and AUX
Command syntax:	M0: no reading is selected M1: Metering of Voltage M2: Metering of Current M3: Metering of Voltage and Current M4: Metering of AUX M5: Metering of Voltage and AUX M6: Metering of Current and AUX M7: Metering of Voltage, Current and AUX
Command response:	> oder !
Query syntax:	M
Query response:	M0: > M1: 01_V:<number> M2: 01_C:< number > M3: 01_V:< number >_C:< number > M4: 01_X:< number > M5: 01_V:< number >_X:< number > M6: 01_C:< number >_X:< number > M7: 01_V:< number >_C:< number >_X:< number >
Default:	M3
Description:	With this command voltage, current and AUX can be metered. The measurement mode can be changed at any time with commands M0 to M7. The first two characters in the response string mark the channel number, which is always „01“ if a power supply with only one output is used. <number> represents the actual metered value and is formatted as floating point number without exponent.
Example:	Response:
M6	>
M	01_C:3.12230000_X:47.34560000

Mess	Measurement dump to the RS232 interface
Type:	Command
Use:	Reading of measurement values continously
Command syntax:	Mess
Response:	> or !
Description:	The measured values of voltage, current and/or AUX are written continously to the serial port. The adjustment of the output form is done with command M0-M7 like before.
Example:	Output:
M3	>
Mess	>
	continously on RS232:
	01_V:9.22530000_C:62.12840000

N	NULL
Type:	Command
Use:	Disabling the output voltage
Command syntax:	N
Response:	> or !
Description:	This command disables the output voltage of the power supply, without changing the reference value. The command “X” enables the voltage in return.

Example:

V17.875X	output voltage = 17.875 Volt
N	output voltage = 0 Volt
X	output voltage = 17.875 Volt

O	Writing / Reading digital outputs
Type:	Command or Query
Use:	Writing / Reading digital outputs x: DOUT1 – Switch the Power Stage ON(0) / OFF(1) y: DOUT2 – reset OVP z: DOUT3
Command syntax:	Oxyz
Response:	O_xyz
Query syntax:	O
Response:	O_xyz

Example:

O111	Response: O_111 - All digital outputs were set to HIGH
O	Response: O_111

Example: Resetting the overvoltage protection (OVP)

L20X	>	// OVP is set to 20 volts
V30X	>	// Set output voltage to 30 volts
I	I_100	// OVP has tripped
L48X	>	// Set OVP to 48 volts
O010;O000	O_010,O_000	// Reset OVP
I	I_000	// OVP is resetted

P Nominal Voltage / Nominal Current

Type: Command or Query
Use: Reading the nominal voltage and current of the power supply
x: nominal voltage
y: nominal amps

Command syntax: Px y
Note: Nominal voltage and current are stored like a parameter in the non-volatile FLASH ROM of the device. To prevent from overwriting the parameters accidentically, the keyword „SERVICE“ has to be transmitted before using this command, which will disable the write-protection of the FLASH area.
Response: P_x_y

Query syntax: P
Response: P_x_y

Example:
P
Response: *P40.000000_25.000000*
- nominal values: 40V/25A

SERVICE
Response: >
P100 10
Response: *P100.00000_10.000000*
- The device was reprorammed to the nominal values of 100V/10A.

Q SRQ masking

Type: Command or Query
Use: Masking the genertion of a SRQ to OV and / or CC
Command syntax: Q00: no SRQ
Q01: SRQ only when CC (current limit)
Q10: SRQ only when OV
Q11: SRQ when CC orOV
Query syntax: Q
Response: 01_Q_00
01_Q_01
01_Q_10
01_Q_11
Default: Q00

Description: With this command a serial request (SRQ) can be allowed if an overvoltage (OV) or a current limit (CC) occurs. The first two characters in the response string mark the channel number, which is always „01“ if a power supply with only one output is used.

Example:
&1
Q10
Response:
&_1 // status byte is cleared // SRQ is allowed
01_Q_10 // SRQ wen an overvoltage condition occurs

V	Voltage
Type:	Command
Use:	Setting the output voltage
Command syntax:	V<number>
Response:	> or !
Default:	V0
Description:	This command sets the output voltage to the desired value. **Cautiont:** Consider percentage / float mode !
Example: V30.83X	float mode: Set overvoltage protection to 30.83 V. percentage mode: Set overvoltage protection to 30.83 % of nominal voltage.

W	Status Byte
Type:	Query
Use:	Reading the Status Byte
Query Syntax:	W
Response:	01_W_<Bit 7..0>
Description:	With this command the status byte can be read. The first two characters in the response string mark the channel number, which is always „01“ if a power supply with only one output is used.
Example: V33.70X	Response: >
W	01_W_00001010 Which means, that the power supply is used with current limit at the moment (see chapter 7.4)

X	EXECUTE
Type:	Command
Use:	Enable programmed values
Command syntax:	X
Response:	> or !
Default:	V0
Description:	The programmed voltage and current are not enabled, before a „X“ is received.. The command „N“ is cancelled by „X“.
Example: V0X	Response: >

Y	Error inquiry
Type:	Query
Use:	Reading error codes from the error queue
Query syntax:	Y
Response:	<errorcode, description>
Description:	This command reads the error which occured at first. If no error occured, the command reads „0, no error“ (see chapter 8.5).
Example:	Response:
Y	-102 // first error: syntax error
Y	-222 // second errorr: data out of range
Y	0 // no more errors left

&	Status Byte, SRQ CLEAR
Type:	Command or Query
Use:	Enables or disables the generation of serial requests. The status byte is resetted when using this command.
Command syntax:	&0: Reset status byte and disable following SRQs. &1: Reset status byte and enable following SRQs.
Query syntax	&
Response:	&_0: SRQs are disabled &_1: SRQs are enabled
Default:	&1
Description:	This command globally enables (&1) or disables (&0) serial requests of the power supply. &0 and &1 reset the status byte to 0.
Example:	Response:
W	01_W_00001100
&1	&_1 // Statusbyte is cleared, SRQ is enabled
W	01_W_00000000