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

BEDIENUNGSHANDBUCH
Option 34/1

Interface IEEE / RS232

Inhaltsverzeichnis

1 Hinweise

1.1 System

Die Option 34 ist ein Interface, das die Fernsteuerung von Spannung, Strom und Überspannungsschutz der Stromversorgung von einem Rechner aus ermöglicht. Zusätzlich können die augenblicklichen Werte von Spannung, Strom und einer dritten Größe (AUX) gemessen werden.

Die Stromversorgung unterstützt die SCPI-Version 1999.0 (Standard Commands for Programmable Instruments). Der SCPI-Standard baut auf dem Standard IEEE 488.2 auf und hat eine Vereinheitlichung der gerätespezifischen Befehle, der Fehlerbehandlung und der Status-register zum Ziel. In Kapitel 4 ist eine Einführung zu SCPI und den entsprechenden Statusregistern angegeben. Die komplette Befehlsliste ist in Kapitel 5 zu finden.

1.2 Wahl der Programmiersyntax

Mit Jumper J4 lässt sich die Option 34 auf die früher verwendeten TET-spezifischen Befehle zurücksetzen. Dieser Befehlssatz ist hersteller-abhängig und weder konform zu SCPI noch zu IEEE 488.2.

Weil jedoch bei Anwendern zahlreiche Programme existieren, die den bisherigen Befehlssatz der Option 34 implementiert haben, und daher ein Umschreiben der Software auf SCPI unter Umständen viel Aufwand bedeuten würde, werden diese Befehle weiterhein unterstützt (siehe Bild 1).

Dabei ist zu beachten, dass SCPI- und TET-Befehle nicht gleichzeitig benutzt werden können. Jumper J4 entscheidet, welche Syntax benutzt werden soll:

J4 offen:	SCPI-Syntax
J4 geschlossen:	TET-Syntax

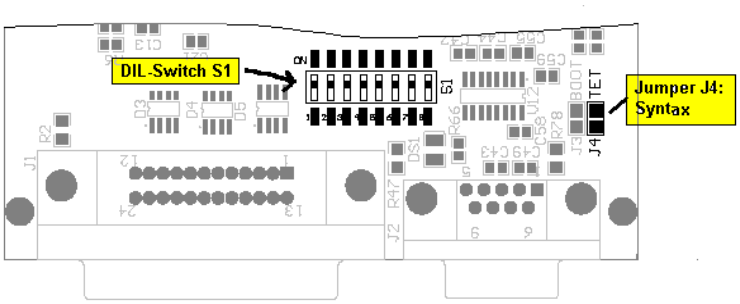


Bild 1: Anordnung von Jumper J4 und DIL-Switch S1

1.3 Umstellen auf Fernbedienung

Nach dem Einschalten wird am Display für etwa 2 Sekunden die momentan eingestellte IEEE488-Adresse angezeigt, gefolgt von Nennspannung und Nennstrom des Netzteils.

Anschliessend ist das Gerät betriebsbereit und zeigt die tatsächlich am Ausgang anstehende Spannung und Strom an.

Das Gerät befindet sich nach dem Einschalten im manuellen Betriebs-zustand („LOCAL“) und kann über die Frontplatte bedient werden.

Die Umstellung auf Fernbedienung („REMOTE“) erfolgt

- bei aktivem IEEE488-Bus, sobald das Gerät von einem Talker einen adressierten Befehl empfängt,
- bei aktiver RS232-Schnittstelle, sobald das Gerät von einem Steuerrechner den Befehl @REM empfängt (bzw. Befehl „B1“ bei TET-Syntax),
- durch den Schalter „INT/EXT“ auf der Frontplatte der Strom-versorgung.

Nach dem Übergang auf „REMOTE“ sind folgende Einstellungen vorhanden:

- Spannungssollwert = 0 V
- Stromsollwert = 0 A
- Überspannungsschutz = 120% der Nennspannung

Das Gerät bleibt im Zustand „REMOTE“, bis es manuell oder über die Fernbedienungsschnittstelle wieder in den manuellen Betriebszustand versetzt wird.

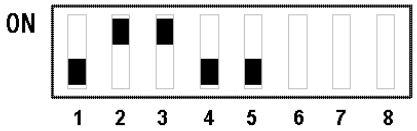
2 IEEE488 Schnittstelle

Die Einstellung der IEEE488 und RS232–Parameter erfolgt über den DIL-Schalter S1 auf der Platine (siehe Bild 1).

2.1 IEEE488 – Geräteadresse einstellen

Die IEEE488–Geräteadresse wird mit den DIL-Schalter 1 bis 5 eingestellt. Dabei gilt das Binärsystem mit

DIL-Schalter	1	2	3	4	5
Wertigkeit	1	2	4	8	16



Beispiel: Für die Geräteadresse 6 wird DIL-Schalter 2 und 3 auf ON gesetzt.

Bei der Auslieferung ist die ***Adresse auf 2*** eingestellt.

Gültige IEEE488-Adressen liegen im Bereich 0 – 30, wobei Adresse 0 häufig von Buscontrollern verwendet wird. Adresse 31 ist als IEEE488-Geräteadresse nicht erlaubt.

2.2 IEEE488-Endecodes

Das Gerät bietet die Möglichkeit, aus 9 verschiedenen IEEE488-Endecodes auszuwählen. Das Gerät erwartet den Abschluss einer IEEE488-Message mit dem eingestellten Endecode bzw. Daten vom Gerät werden mit diesem Endecode terminiert.

Bei der Auslieferung wird der Endecode EOI + LF benutzt.

2.2.1 EOI (End or Identify)

Soll als Endecode nur die Steuerleitung EOI benutzt werden, dann wird DIL-Schalter 6 auf ON und DIL-Schalter 8 auf OFF gesetzt.

Soll ein anderer Endecode verwendet werden, so wird DIL-Schalter 6 und 8 auf OFF gesetzt (siehe Tabelle 1). Der Endecode muss dem Gerät dann in einem Programmierlauf mitgeteilt werden (Kapitel 2.2.2).

DIL-Schalterstellung	Endecode
	EOI
	Programmierter Endecode (siehe 2.2.2)

Tabelle 1: Auswahl zwischen EOI und programmiertem Endecode

2.2.2 Programmierter Endecode

Um einen Endecode ins Gerät einzuprogrammieren, ist folgender Ablauf notwendig:

1. Mit den DIL-Schaltern 1 bis 5 die unerlaubte IEEE488-Adresse 31 einstellen. Mit den DIL-Schaltern 6 bis 8 den gewünschten Endecode nach Tabelle 2 einstellen.
2. Gerät einschalten und warten, bis REM-LED und Display im ½-Sekundentakt blinken. Der Endecode ist dann gespeichert.
3. Gerät ausschalten, eine erlaubte IEEE488-Adresse (siehe 2.1) einstellen, DIL-Schalter 6 und 8 auf OFF setzen. Der Endecode wird nun ab dem nächsten Einschalten verwendet.

DIL-Schalterstellung	Programmierter Endecode
	EOI + CR
	EOI + CR + LF
	EOI + LF
	EOI + LF + CR
	CR
	CR + LF

	LF
	LF + CR

Tabelle 2: programmierbare Endecodes

2.3 SRQ (Service Request)

Bei Verwendung der TET-Syntax lässt sich mit DIL-Schalter 7 ein Service Request (SRQ) erlauben (ON) oder verbieten (OFF). Siehe Tabelle 3.

DIL-Schalterstellung	
	SRQ erlaubt
	SRQ verboten

Tabelle 3: SRQ-Einstellung bei TET-Syntax

Bei Verwendung der SCPI-Syntax wird DIL-Schalter 7 nicht verwendet. Die Erzeugung eines SRQ wird nach IEEE 488.2 über das Register SRE (Service Request Enable) vorgenommen (siehe Kapitel 4).

2.4 IEEE-Schnittstellenfunktionen

SH1	-Source-Handshake-Fähigkeit
AH1	-Acceptor-Handshake-Fähigkeit
T5	-Talker-Fähigkeit (Basic Talker, Serial Poll, unadressed on MLA, Send END or EOS)
L3	- Listener-Fähigkeit (Basic Listener, unadressed on MTA, Detect END or EOS)
SR1	- Service-Request-Fähigkeit
RL1	- Remote-/Local-Fähigkeit
PP1	- Remote-Parallel-Poll-Konfiguration
DC1	- „DeviceClear“-Fähigkeit
DT0	- keine Triggerfähigkeit
C0	- keine Controller-Fähigkeit

3 RS232-Schnittstelle

3.1 Verdrahtung

Bei serieller Kommunikation ist ein Kabel mit den Signalen RxD, TxD und GND notwendig. Hardwarehandshake wird nicht unterstützt. Es eignet sich ein handelsübliches, 1:1 belegtes Schnittstellenkabel.

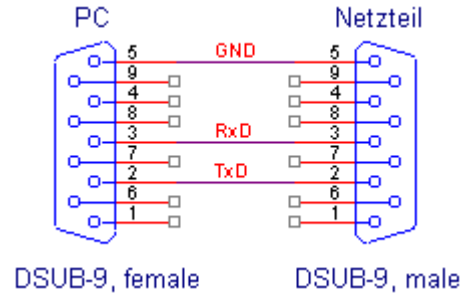


Bild 2: serielles Kabel

3.2 RS232-Schnittstellenparameter

Bei Auslieferung ist die serielle Schnittstelle auf folgenden Parameter eingestellt:

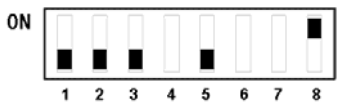
- 9600 Baud
- 1 Startbit
- 8 Datenbit
- 1 Stopbit
- kein Paritybit
- Endecode = Terminal

Endecode „Terminal“ bedeutet, dass von der PC-Seite ein Terminalprogramm, wie z.B. „HyperTerminal“ unter Windows, zur Fernsteuerung verwendet werden kann.

Einstellen der RS232-Schnittstellenparameter

Um die RS232-Parameter zu ändern, muss mit DIL-Schalter S1 eine Umprogrammierung vorgenommen werden:

1. Mit DIL-Schalter 5 auf OFF und 8 auf ON die Umprogrammierung eingeschalten.
2. Mit DIL-Schalter 1 bis 3 die Baudrate einstellen (siehe Tabelle 4)
3. Mit DIL-Schalter 4 den Übertragungsrahmen einstellen (siehe Tabelle 5)
4. Mit DIL-Schalter 6 und 7 den RS232-Endecode einstellen (siehe Tabelle 6)
5. Gerät einschalten und warten, bis REM-LED und Display im ½-Sekunden-Takt blinken. Die Parameter sind nun gespeichert.
6. Gerät ausschalten, DIL-Schalter 8 auf OFF setzen und wieder die IEEE488-Adresse (Kapitel 2.1, 2.2) einstellen.
7. Ab dem nächsten Einschalten arbeitet das Gerät mit den programmierten RS232-Parametern.

DIL-Schalterstellung	Baudrate
	1200

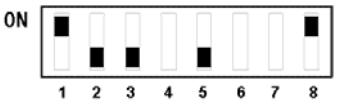
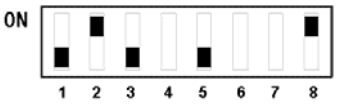
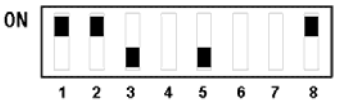
DIL-Schalterstellung	Baudrate
	2400
	4800
	9600
	19200
	38400
	57600
	115200

Tabelle 4: Baudrateneinstellung

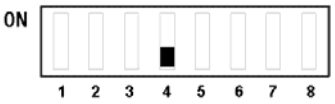
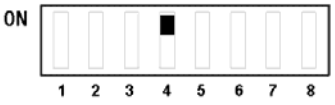
DIL-Schalterstellung	Frame
	1 Startbit, 8 Datenbit, 1 Stopbit
	1 Startbit, 8 Datenbit, 1 Stopbit, 1 Paritybit (even)

Tabelle 5: Übertragungsrahmen einstellen

DIL-Schalterstellung	RS232-Endecode
	LF (0x0A)
	CR (0x0D)
	^Z (Ctrl-Z) (0x1A)
	Terminal (CR/LF, CR vom Terminal)

Tabelle 6: RS232-Endecode-Einstellung

4 Fernbedienen mit SCPI

Das Gerät unterstützt die SCPI-Version 1999.0 (Standard Commands for Programmable Instruments). Die SCPI-Spezifikation definiert eine Programmiersprache, die zum Steuern von Test- und Messgeräten benutzt wird.

SCPI baut auf IEEE 488.2 auf. Der IEEE 488.2-Standard definiert das Übertragungsprotokoll und die sogenannten „Common Commands“, die von allen 488.2-kompatiblen Geräten unterstützt werden. SCPI legt zusätzlich die Befehle fest, die für einen bestimmten Typ von Instrument notwendig sind, z.B. Oszilloskope, Funktionsgeneratoren etc.

Der Vorteil von SCPI ist die Austauschbarkeit zwischen verschiedenen Geräten und Geräteherstellern. Derselbe Befehl, der zum Beispiel eine Spannungsmessung auf einem Multimeter durchführt, führt auf einem anderen Gerät, zum Beispiel einem Power-Meter, exakt die gleiche Funktion aus.
Ein Steuerprogramm, geschrieben für das programmierbare Netzteil des einen Herstellers, wird mit wenig Änderungen mit dem Netzteil von eines anderen Hersteller funktionieren.

4.1 Befehlssyntax

SCPI ist eine Erweiterung der 488.2-Spezifikation bezüglich der Datenformate, der Anwendung von „Common Commands“ und dem 488.2-Statussystem. Wie in 488.2 gibt es „program messages“, die als Daten vom Controller zum Instrument geschickt werden und entsprechend „response messages“, die als Anwort vom Instrument zum Controller übertragen werden.

Wie in 488.2 definiert SCPI sowohl Befehle als auch Abfragen. In SCPI gibt es zu jedem Befehl, der z.B. einen Wert setzt, eine passende Abfrage, die diesen Wert zurückliest. Ausnahmen werden in der Dokumentation aufgeführt.

4.1.1 „Common Commands“-Syntax

„ Common Commands“ sind geräteunabhängige Befehle, die von jedem 488.2- und damit auch von jedem SCPI-Gerät verstanden werden. „Common Commands“ bestehen aus einem Header, dem ein `'*'` vorausgestellt ist und eventuell einem Parameter.

Beispiele:

*RST	Gerät zurücksetzen
*SRE 48	Bits 4 und 5 des „Service Request Enable“-Registers setzen
*SRE #H30	wie *SRE 48 (48dezimal = 30hex)
*SRE #B00110000	wie *SRE 48 (48dezimal = 00110000bin)
*SRE?	SRE-Register abfragen

4.1.2 SCPI-Syntax

SCPI organisiert Befehle in verschiedene Abschnitte, den sogenannten „Subsystems“. Zum Beispiel beinhaltet das Subsystem SOURCE Befehle, bei denen das Instrument als Quelle physikalische Größen ausgibt. Bei einer DC-Stromversorgung sind dies Spannung und Strom, bei einem Funktionsgenerator können z.B. zusätzlich Frequenz und Kurvenform definiert werden.

Baumstruktur

Die Befehle jedes Subsystems sind in einer hierarchischen Struktur angeordnet ähnlich dem hierarchischen Dateisystem, das auf Computern zu finden ist.

Als Beispiel ist das SOURCE-Subsystem in Bild 3 dargestellt.

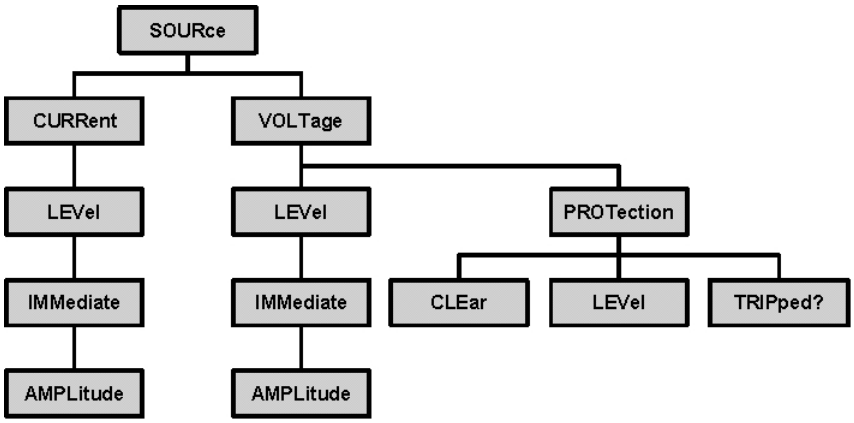


Bild 3: Befehlsbaum des Subsystems „SOURCE“

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 `':'`.

Zum Beispiel setzt der Befehl

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

den Ausgang des Netzteils auf 100 Volt.

Pfad

- Ein Message-Terminator setzt den aktuellen Pfad auf „root“. Als Message-Terminator wird gewöhnlich <LF>, EOI oder <LF> + EOI benutzt.
- Nach dem Einschalten oder wenn der *RST-Befehl geschickt wurde, wird der aktuelle Pfad ebenfalls auf „root“ gesetzt.
- Ein Doppelpunkt (':')ist das Trennzeichen für die Ebenen in der Hierarchie. Jedesmal, wenn der Befehlsparser einen Doppelpunkt im Befehlsstring findet, bewegt er sich in der Hierarchie um eine Ebene nach unten.
- Ein Strichpunkt (';') trennt zwei Befehle im selben Befehlsstring ohne den aktuellen Pfad zu ändern.

Beispiel:

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

ist dasselbe wie

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

optionale Schlüsselwörter

Schlüsselwörter, die in eckigen Klammern angegeben sind, können weggelassen werden. Die Befehlszeilen verkürzen sich dadurch zum Teil erheblich. Optionale Schlüsselwörter werden aus Kompatibilitäts-gründen zum SCPI-Standard unterstützt.

Beispiel: Der Befehl für das Einstellen eines Stromes ist im SOURce-Subsystem folgendermaßen definiert (siehe Kapitel 5):

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

Um einen Strom von 10A auszugeben (bzw. die Strombegrenzung auf 10A zu setzen), kann daher folgender Befehl verwendet werden:

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

Oder, wenn die Schlüsselwörter in eckigen Klammern weggelassen werden, kurz:

```
CURRent 10A
```

Lang- und Kurzform

Die meisten Schlüsselwörter werden in SCPI-Schreibweise als Strings mit Großbuchstaben, gefolgt von Kleinbuchstaben, angegeben. Die Großbuchstaben kennzeichnen dabei die Kurzform des Befehls. Die Kleinbuchstaben geben die Langform an und können, falls gewünscht, weggelassen werden.

Zum Beispiel ist

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

dasselbe wie

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

Die Groß- und Kleinschreibung dient dabei nur der Kennzeichnung von Lang- und Kurzform. SCPI selbst unterscheidet nicht zwischen Groß- und Kleinschreibung, zum Beispiel ist

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

Parameter

- Whitespace-Zeichen, wie z.B. <tab> oder <space> werden vom Parser ignoriert. Whitespace-Zeichen in einem Befehlsschlüsselwort sind jedoch nicht erlaubt. Zum Beispiel ist SOUR ce kein gültiger Befehl.
- Whitespace-Zeichen sind erforderlich um einen Parameter vom vorausgehenden Befehl zu trennen. Zum Beispiel führt „ SOURce:VOLTage40.5V“ zu einem Syntaxfehler, man muss „ SOURce:VOLTage 40.5V“ schreiben.
- Kommas (',') trennen mehrere Parameter in einem Subsystem-Befehl.
- „ Common Commands“, wie z.B *RST, sind keine Subsystem-Befehle und werden nicht als Teil eines Pfades interpretiert.

Einheiten

⑩ SCPI unterstützt das Verwenden von Einheiten. Wird keine Einheit angegeben, ist die Grundeinheit gemeint. Zum Beispiel sind folgende Befehle identisch:

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

Zahlenwerte

Gebräuchliche Zahlenwerte:	Zahlenwerte werden in gebräuchlicher Form eingege-ben, also mit Vorzeichen, Dezimalpunkt und evtl. Exponent, z.B.: `12.0, -5.34, 1000E-3`. Bei physikalischen Größen kann die Einheit angegeben werden, z.B. `100mA, 50V`.
MIN/MAX	`MINimum`: 0 Volt bzw. 0 Ampere `MAXimum`: Nennspannung bzw. Nennstrom
DEF	`DEFault` bezeichnet einen Wert, der der Grundein-stellung des Geräts entspricht, wie bei einem *RST.
Boolesche Parameter	`ON` und `OFF` Zum Beispiel `OUTPut ON` Bei einer Abfrage wird `0(OFF)` oder `1(ON)` zurück-geliefert.

4.2 Befehlsreihenfolge

Alle Befehle werden der Reihe nach ausgeführt, so wie sie über die Schnittstelle eintreffen. Das vollständige Ausführen einer Geräte-funktion ist jedoch nicht zwingend notwendig, bevor der nächste Befehl gestartet wird.

Zum Beispiel wird die Stromversorgung mit „ *SOURce:VOLTage 100*“ aufgefordert, 100 Volt am Ausgang einzustellen. Während der Ausgang noch auf die geforderte Spannung einschwingt, wird bereits der nächste Befehl bearbeitet. Der *SOURce:VOLTage*-Befehl und der nachfolgende Befehl werden daher überlappend (parallel) ausgeführt.

Dieses Verhalten ist nicht immer wünschenswert, weil in manchen Fällen die Spannung der Stromversorgung eingeschwungen sein muss, bevor der nächste Befehl ausgeführt wird. Deshalb kann mit den nachfolgend beschriebenen ***Synchronisierbefehlen*** ein sequentielles Verhalten erzwungen werden.

In diesem Gerät werden bis auf das *SOURCE*-Subsystem alle Befehle sequentiell verarbeitet. Nur das *SOURCE*-Subsystem, das zum Einstellen von Spannung und Strom dient, wird überlappend verarbeitet. Eine sequentielle Ausführung kann mit den Befehlen **OPC*, **OPC ?* und **WAI* *erzwungen werden.*

4.2.1 *OPC – Operation Complete

Wird der Befehl **OPC* angegeben, dann setzt das Gerät das Bit „Operation Complete“ (*OPC,* Bit 0) im Standard-Event-Status-Register auf *TRUE*, wenn alle zuvor angegebenen Befehle vollständig ausgeführt wurden. **OPC* sollte als letzter Befehl in einer terminierten Befehlsfolge benutzt werden. Der **OPC*-Mechanismus wird bei einem *power-on*, einer Device-Clear-Message (*DCL*), einem **RST* und einem **CLS* zurückgesetzt.

Beispiel: *SOURce:VOLTage 100;*OPC*
Das *OPC*-Bit wird gesetzt, sobald die Ausgangs-spannung stabil innerhalb der Toleranz (0,5% Nennspannung)an den Ausgangsklemmen anliegt (*oder sobald die Strombegrenzung erreicht ist*).

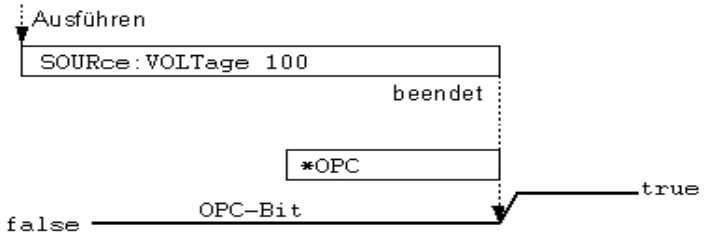


Bild 4: *OPC-Befehlsausführung

4.2.2 *OPC? – Operation Complete Query

Wird die Abfrage **OPC?* angegeben, dann schreibt das Gerät das ASCII-Zeichen „*1*“ (31hex) in den Ausgabepuffer, sobald alle zuvor angebegebenen Befehle ausgeführt wurden. **OPC?* sollte als letzter Befehl in einer terminierten Befehlsfolge benutzt werden.

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

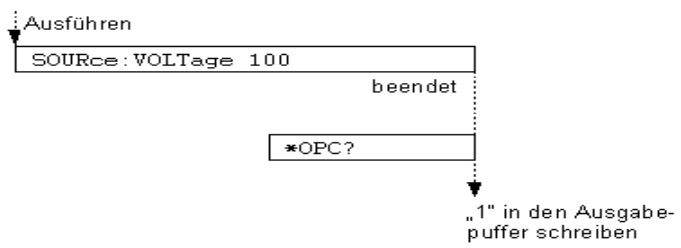


Bild 5: *OPC?-Befehlsausführung

4.2.3 *WAI – Wait-to-Complete

Wird der Befehl **WAI* angegeben, dann wartet das Gerät mit der Abarbeitung folgender Befehle, bis alle zuvor angegebenen Befehle abgearbeitet wurden.

Beispiel: *SOUR:VOLT 100; *WAI; :MEAS:CURR?*
Gibt 100 Volt aus und wartet, bis die Spannung eingeschwungen ist (oder die Strombegrenzung erreicht wird). Dann wird der Strom gemessen.

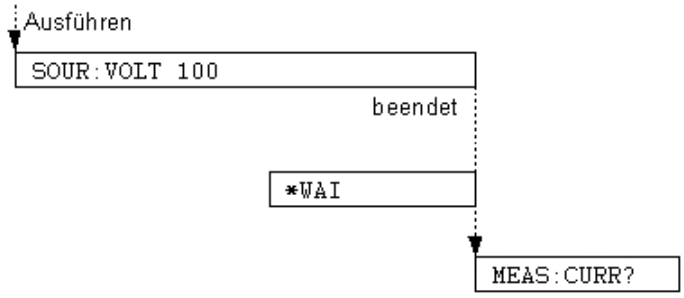


Bild 6: *WAI-Befehlsausführung

4.3 Status-System

Mit Hilfe des Status-Systems kann der Anwender Informationen über den momentanen Gerätezustand abrufen und entsprechend reagieren. Ist während der Ausführung des Anwenderprogramms zum Beispiel ein Fehler aufgetreten oder hat ein anderes Ereignis stattgefunden, wird dieser Vorgang in einem Register des Status-Systems aufgezeichnet.

Welche Zustände und Ereignisse berichtet werden, kann vom Anwender beliebig angepasst werden, indem Masken-Bits in Enable-Registern gesetzt werden.

Das Status-System ist streng hierarchisch aufgebaut. An oberster Stelle befindet sich das Status Byte. Das Status Byte wird aus den Summenbits der unterlagerten Statusregister gebildet (siehe Bild 7).

Service Request

Geräte mit IEEE488-Interface können dem Anwendungsprogramm ihren Gerätezustand signalisieren, indem sie die sogenannte Service-Request-Line (SRQ) auf LOW ziehen. Der Systemcontroller, gewöhn-lich ein PC mit IEEE488-Controller-Karte, erkennt am SRQ, dass ein Gerät Service anfordert. Der Controller fragt anschliessend die Geräte am Bus nacheinander mittels „Serial Poll“ ab. Die Geräte antworten auf einen „Serial Poll“ mit ihrem Status Byte. Das Gerät, das den SRQ verursacht hat, ist an dem gesetzten Bit 6 (RQS, „Request Service“) zu erkennen.

Indem man Bits im Status Request Enable Register (SRE) auf logisch 1 setzt, kann man festlegen, welches Ereignis einen Service Request auslösen kann. Mit dem Befehl **SRE* („Service Request Enable“) werden wird das SRE-Register beschrieben.

Zum Beispiel setzt der Befehl **SRE 16* Bit 4 im Enable-Register, das sogenannte MAV-Bit („Message Available). Sobald im Ausgabepuffer Daten vorhanden sind, wird das MAV-Bit gesetzt. Dieses Bit wird einen SRQ auslösen, weil es vorher mit **SRE 16* „enabled“ wurde.

Nach dem Einschalten des Gerätes sind die Enable-Register gelöscht, deshalb müssen die Flags, die einen Service Request erzeugen sollen, explizit auf logisch 1 gesetzt werden.

Das Status Byte, verbunden mit dem Service Request, ist die einzige Möglichkeit, mit der das Gerät ein laufendes Anwendungsprogramm unterbrechen und die Aufmerksamkeit des Controllers auf sich ziehen kann.

4.3.1 Aufbau des Status-Systems

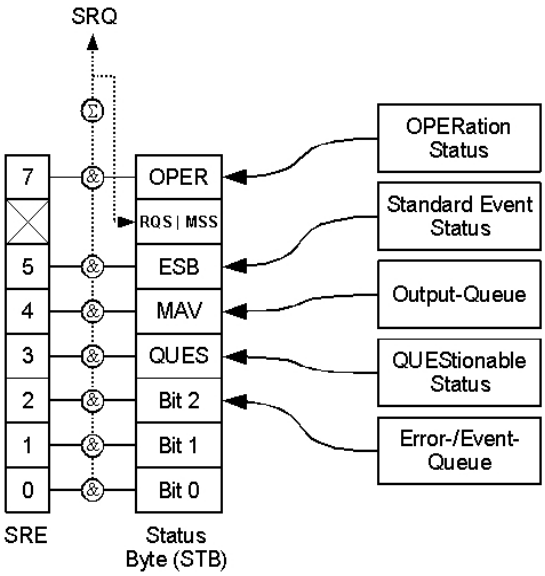


Bild 7: Status Byte

Bild 7 zeigt die Bestandteile des Status-Systems. & stellt ein logisches UND dar, Σ ein logisches ODER.

Dem Status Byte überlagert sind die Status Register

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

und die Queues

- Output-Queue (Ausgabepuffer)
- Error/Event-Queue

Die überlagerten Status-Register werden als Summenbit im Status Byte abgebildet. Wenn Daten in der Output-Queue vorhanden sind, wird das Bit MAV (Message Available) im Status Byte gesetzt. Wenn ein Eintrag in der Error/Event-Queue vorliegt, wird Bit 2 des Status Byte gesetzt.

Soll bei Auftreten eines bestimmten Ereignisses ein Service Request (SRQ) ausgelöst werden, so ist vorher das entsprechende Bit im Service Request Enable Register (SRE) zu setzen.

Tabelle 7 zeigt die Bedeutung der Bits im Status Byte.

Statusbyte:			
Bit-Nr.	Abk.	Name	Bedeutung
7	OPER	OPERation Status	Summenbit der Operation Status Struktur
6	RQS	Requested Service	Zeigt an, dass das Gerät Service angefordert hat. Ein SRQ-Interrupt wurde erzeugt. *Das RQS-Bit wird als Antwort auf einen Serial Poll geschickt,* um dem Controller anzuzeigen, dass der SRQ von diesem Gerät ausgelöst wurde.
	MSS	Master Summary Status	Das MSS-Bit wird als Antwort auf eine **STB?*-Abfrage geliefert. Das MSS-Bit wird gelöscht, wenn die überlagerten Statusregister gelöscht werden bzw. das SRE-Register durch " **SRE 0*" gelöscht wird.

Statusbyte:			
5	ESB	Event Summary Bit	Summenbit des Event Status Registers
4	MAV	Message Available	Zeigt an, das im Ausgabepuffer Daten vorhanden sind, die gelesen werden können.
3	QUES	QUEStionable Status	Summenbit der QUEStionable Status Struktur
2	Error/Event Queue		Zeigt an, dass die Error/Event Queue nicht leer ist.

Tabelle 7: Status Byte

Befehle, die das Statusbyte betreffen:

**STB? - Statusbyte-Abfrage*

Liefert den Inhalt des Statusbytes. An Position 6 wird das MSS-Bit gelesen. Der Inhalt des Statusbytes wird nicht verändert. Der Inhalt ändert sich nur, wenn die überlagerten Statusregister, die als Summen-bits im Status Byte dargestellt werden, gelöscht werden.

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

Setzt das SRE-Register auf den angegebenen Wert. Ein gesetztes Bit verursacht einen SRQ-Interrupt, sobald das entsprechende Ereignis auftritt.

**SRE? – Service Request Enable - Abfrage*

Liefert den Inhalt des SRE-Registers zurück. Der Wert von Bit 6 ist dabei immer 0.

Beispiele:

**STB?* Liefert zum Beispiel den Wert 20 zurück. Dies entspricht binär 00010100_b. Das heisst, dass im Status Byte die Bits 2 (Error/Event-Queue) und 4 (MAV) TRUE sind.

**SRE 12* 12 dezimal entspricht binär 00001100_b. Das Gerät erzeugt daher einen SRQ, wenn in der Error/Event-Queue ein Eintrag vorhanden ist (Bit 2) oder wenn das Summenbit QUES gesetzt ist (Bit 3).

4.3.2 Überlagerte Statusregister und Queues

Wie bereits in Bild 7 gezeigt, gibt es folgende überlagerte Status-register:

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

Jedes dieser Statusregister bildet ein Summenbit, das sich im Status Byte wiederfindet. Über das Summenbit lässt sich bei Bedarf ein SRQ-Interrupt erzeugen.

Zusätzlich wird der Zustand von

- Ausgabepuffer
- Error/Event Queue

im Status Byte abgebildet. Falls ein Eintrag in der jeweiligen Queue vorhanden ist, wird im Status Byte ein Bit gesetzt.

4.3.2.1 Standard Event Status Register

Das Standard Event Status Register wird mit dem Summenbit ESB im Status Byte dargestellt. Die Erzeugung des Summenbits kann mit dem Standard Event Enable Register maskiert werden (siehe Bild 8).

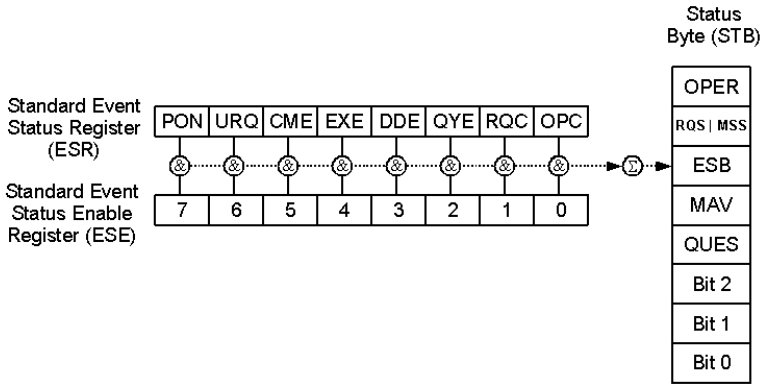


Bild 8: Standard Event Status Register

Standard Event Status Register:			
Bit-Nr.	Abk.	Name	Bedeutung
7	PON	Power ON	Wird beim Einschalten der Netzspannung gesetzt.
6	URQ	User Request	- nicht benutzt -
5	CME	Command Error	Wird gesetzt, wenn ein Syntaxfehler vorliegt. In die Error/Event-Queue wird ein Fehler zwischen -100 und -199 eingetragen (siehe Fehlercodes).
4	EXE	Execution Error	Wird gesetzt, wenn ein Befehl nicht ausgeführt werden kann, weil z.B. der Wertebereich überschritten wird. In die Error/Event-Queue wird ein Fehler zwischen -200 und -299 eingetragen.
3	DDE	Device Dependent Error	Wird gesetzt, wenn ein geräteabhäng-iger Fehler auftritt. In die Error/Event-Queue wird ein Fehler zwischen -300 und -399 oder ein positiver Fehlercode eingetragen.
2	QYE	Query Error	Wird gesetzt, wenn z.B. ein Befehl gesendet wird, ohne die Antwort auf einen vorhergehenden Befehl gelesen zu haben; In die Error/Event-Queue wird ein Fehler zwischen -400 und -499 eingetragen.
1	RQC	Request Control	- nicht benutzt -
0	OPC	Operation Complete	Wird nach einem **OPC* – Befehl gesetzt, wenn alle bisherigen Befehle ausgeführt wurden (siehe auch 4.2 – Befehlsreihenfolge)

Tabelle 8: Bedeutung der Bits im Standard Event Status Register

Befehle, die das Standard Event Status Register betreffen:

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

Liefert den Inhalt des Standard Event Status Registers (ESR). Das Register wird bei der Abfrage gelöscht.

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

Setzt das ESE-Register auf den angegebenen Wert. Ein gesetztes Bit verursacht, dass bei dem entsprechenden Ereignis das Summenbit ESB auf TRUE gesetzt wird.

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

Liefert den Inhalt des ESE-Registers zurück. Der Inhalt des Registers bleibt bei der Abfrage unverändert.

Beispiele:

**ESR?* Liefert zum Beispiel den Wert 21. Dies entspricht binär 00010101_b. Das heisst, dass im ESR-Register die Bits 0 (OPC), Bit 2 (QYE) und Bit 4 (EXE) gesetzt sind. Das ESR-Register wird bei der Abfrage gelöscht.

*ESE 24 24 dezimal entspricht binär 00011000b. Das Summenbit ESB wird daher gesetzt, sobald ein Device Dependent Error (Bit 3) oder ein Execution Error (Bit 4) auftritt.

4.3.2.2 QUEStionable Status

Den Aufbau der QUEStionable Status Register zeigt Bild 9. Die Bedeu-tung der Bits ist in Tabelle 9 dargestellt.

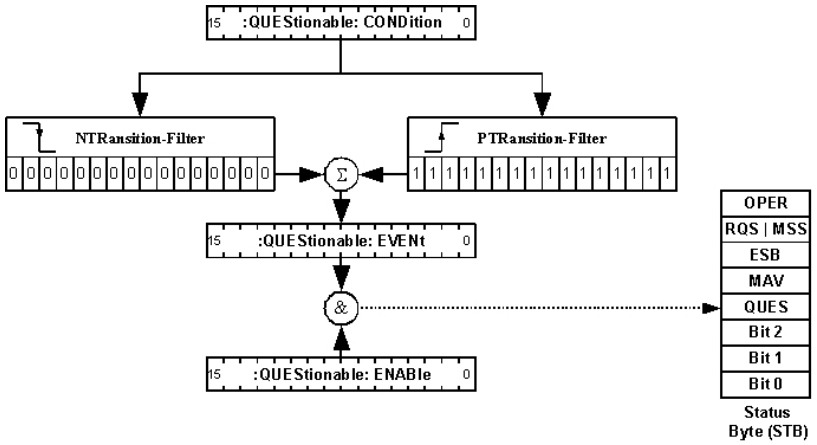


Bild 9: QUEStionable Status Register

STATus:QUEStionable:CONDition - Register			
Bit-Nr.	Abk.	Name	Bedeutung
11 ... 15			- nicht benutzt -
10	WDTR	Watchdog-Timer-Reset	Reset durch Watchdog-Überlauf
9	OVP	Over Voltage Protection	Zeigt einen ausgelösten Überspannungsschutz an.
5 ... 8			- nicht benutzt -
4	TEMP	Temperatur	Wird bei Temperaturfehler gesetzt.
2,3			- nicht benutzt -
1	CURR	CURRent	Wird gesetzt, wenn der Strom ungleich dem Stromsollwert ist (unterhalb der Strombegrenzung).
0	VOLT	VOLTage	Wird gesetzt, wenn die Spannung ungleich dem Spannungssollwert ist.

Tabelle 9: Bits im STATus:QUEStionable Register

CONDition und EVENt-Register sind zu unterscheiden. Das CONDition-Register zeigt einen bestimmten Zustand des Gerätes an. Zum Beispiel zeigt das VOLT-Bit an, das der Ausgang der Stromversorgung nicht auf den Spannungs-Sollwert eingeschwungen ist. Ist der Sollwert erreicht, wird das VOLT-Bit auf 0 gesetzt.

Das EVENt-Register zeichnet auf, wenn sich ein Bit im Condition-Register geändert hat. Wird zum Beispiel der Überspannungsschutz ausgelöst, dann ändert sich das Bit OVP im CONDition-Register von 0 auf 1. Dieses Ereignis wird im EVENt-Register aufgezeichnet.

Das EVENt-Register wird benutzt, um dem Anwender mitzuteilen, dass ein Ereignis stattgefunden hat. Das EVENt-Register wird gelöscht, sobald es mit der Abfrage *STATus:QUEStionable[:EVENt]?* gelesen wird. Im Gegensatz dazu wird das CONDition-Register bei einer Abfrage nicht gelöscht, weil es nur vom Gerätezustand abhängig ist.

Transition-Filter sind Bestandteil des SCPI-Statussystems. Sie wirken als Flankendetektor auf das CONDition-Register. Die Änderung eines Bit im CONDition-Register wird durch den Transition-Filter erkannt und im EVENt-Register angezeigt.

Die Transition-Filter sind in diesem Gerät nicht programmierbar. Im PTRansition-Filter sind alle Bit fest auf „1“ gesetzt, im NTRansition-Filter alle Bit gelöscht. Das heisst, im EVENt-Register werden nur Ereignisse aufgezeichnet, die sich im CONDition-Register von 0 auf 1 ändern, jedoch nicht von 1 auf 0. Die aufgezeichneten Bits im EVENt-Register bleiben solange gesetzt, bis das EVENt-Register vom Anwender abgefragt wird.

Beispiele:

STATus:QUEStionable?	Gibt den Inhalt des QUEStionable:EVENt–Registers zurück
STAT:QUES:ENAB 528 *oder* *STAT:QUES:ENAB #H210*	Dezimal 528 entspricht binär 0000 0010 0001 0000b. Wenn im EVENt-Register Bit 4 (Temperatur) oder Bit 9 (Überspannungsschutz) auftritt, dann wird das Summenbit QYE im Status Byte gesetzt.
STAT:QUES:CONDition?	Fragt das QUEStionable-CONDition-Register ab.

4.3.2.3 OPERation Status

Die OPERation Status Register sind genauso aufgebaut wie die QUEStionable Status Register. Das Gerät benutzt allerdings keine Bits im *STATus:OPERation:CONDition* – Register.

Aus Konformitätsgründen mit dem SCPI-Standard kann das Register gelesen werden Eine Abfrage mit *STATus:OPERation:CONDition?* liefert immer 0 zurück.

4.3.2.4 Ausgabepuffer (Output-Queue)

Das Gerät hält Antworten solange im Ausgabepuffer, bis sie vom Controller gelesen werden. Wenn Daten im Ausgabepuffer vorhanden sind, wird dies durch das Bit MAV (Message Available) im Status Byte angezeigt.

Wird das Gerät als Talker adressiert, ohne dass Daten im Ausgabe-puffer vorhanden sind, wird der Fehler „QUERY UNTERMINATED“ erzeugt. Das Gerät schickt keine Daten und der Controller wird einen Timeout erleiden.

Empfängt das Gerät bereits die nächste Message, obwohl noch Daten im Ausgabepuffer vorhanden sind, wird der Fehler „QUERY INTER-RUPTED“ erzeugt. Der Ausgabepuffer wird gelöscht und der nächste Befehl bearbeitet.

4.3.2.5 Error/Event-Queue

Fehler und Ereignisse, die innerhalb des Gerätes auftreten, werden in den oben beschriebenen Status- und EVENt-Registern als Bit aufgezeichnet. Diese Bit-Werte bleiben erhalten, bis das EVENt-Register vom Controller gelesen wird oder bis ein **CLS* (Clear Status) – Befehl empfangen wird.

In welcher Reihenfolge Fehler und Ereignisse aufgetreten sind, kann aus den Statusregistern jedoch nicht ermittelt werden. Dazu dient die Error/Event-Queue, die eine zeitliche Abfolge der aufgetretenen Fehler und

Ereignisse bereithält. Die Error/Event-Queue ist ein FIFO-Puffer (First-In, First-Out). Das bedeutet, dass beim Lesen der Queue mit *SYSTem:ERRor?* der Fehler gelesen wird, der zuerst auftrat.

Jeder Aufruf von *SYSTem:ERRor?* liefert einen Eintrag aus der Error-/Event-Queue. Der Aufruf kann solange wiederholt werden, bis keine Fehlermeldungen mehr in der Queue vorhanden sind. Das Gerät antwortet dann mit "*0, No error*".

Liegt ein Eintrag in der Error/Event-Queue vor, wird Bit 2 im Status Byte gesetzt. Das Bit kann bei Bedarf zum Erzeugen eines SRQ verwendet werden.

4.3.3 Parallel Poll

Selten benutzt wird das sogenannte Parallel-Polling, bei dem der Status mehrerer Geräte gleichzeitig abgefragt werden kann. Das Gerät liefert beim Parallel-Poll das „ist-Flag" (individual status Flag) auf einer Datenleitung zurück. Weil diese Statusinformation nur 1 Bit umfasst und 8 Datenleitungen zur Verfügung stehen, können 8 Geräte gleichzeitig gepolled werden.

Jedes Gerät, das an einem Parallel Poll teilnehmen soll, muss vorher konfiguriert werden. Dem Gerät wird dabei mitgeteilt, auf welche Datenleitung (1 bis 8) das ist-Flag gelegt werden soll und ob die Datenleitung bei „ist=TRUE" auf 1 oder 0 gesetzt werden soll.

Bild 10 zeigt, wie das ist-Flag gebildet wird. Das Statusbyte (STB) wird mit dem PRE-Maskenregister „verUNDet". Das Ergebnis dieser UND-Verknüpfung erscheint als ist-Flag.

Das PRE-Register kann mit dem Befehl **PRE <0..255>* beschrieben werden. Das ist-Flag lässt sich neben dem Parallel Poll auch mit dem Abfragebefehl **IST?* auslesen.

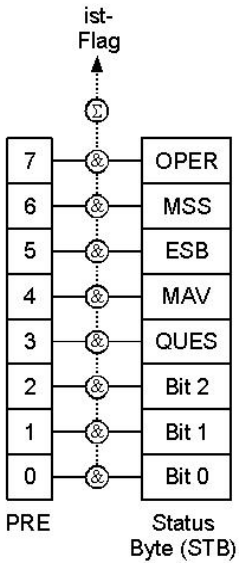


Bild 10: Bildung des ist-Flags

5 SCPI- Befehle

In diesem Kapitel werden alle im Gerät implementierten Befehle beschrieben. Die Schreibweise entspricht dem SCPI-Standard Version 1999.0.

Jeder Befehl bzw. jedes Befehls-Subsystem wird erst in einer Tabelle mit den zugehörigen Parametern und einem Kommentar dargestellt und anschliessend detailliert beschrieben.

Lang- und Kurzform von Befehlen

Die Kurzform eines Befehles oder Schlüsselwortes wird in der Beschreibung mit Großbuchstaben gekennzeichnet. Das Gerät selbst unterscheidet jedoch nicht zwischen Groß- und Kleinschreibung.

Befehl oder Abfrage

Für jeden Befehl gibt es in der Regel eine zugehörige Abfrageform, indem ein „?“ an den Befehl gehängt wird.

Bei Befehlen, die keine Abfrageform besitzen, und bei Befehlen, die nur als Abfrage gestellt werden, wird dies als Bemerkung in der rechten Spalte der Tabelle angegeben.

5.1 Common Commands

Common Commands sind in allen SCPI-Geräten vorhanden. Im Standard IEEE488.2 werden diese Befehle als notwendig voraus-gesetzt. Viele Common Commands betreffen das Statussystem, welches in Kapitel 4 beschrieben ist.

Befehl	Parameter	Bemerkung
***CLS**		Clear Status keine Abfrage
***ESE**	0 ... 255	Event Status Enable
***ESR?**		Standard Event Status Query nur Abfrage
***IDN?**		Identification Query nur Abfrage
***IST?**		Individual Status Query nur Abfrage
***OPC**		Operation Complete
***PRE**	0 ... 255	Parallel Poll Register Enable
***RST**		Reset keine Abfrage
***SRE**	0 ... 255	Service Request Enable
***STB?**		Read Status Byte Query nur Abfrage
***TST?**		Self Test Query nur Abfrage
***WAI**		Wait-to-Continue keine Abfrage

5.1.1 *CLS

Clear Status löscht alle Event-Register und die Error/Event-Queue. Das heisst, es werden alle Ereignisse aus dem Statussystem entfernt, die von der Anwendung noch nicht abgefragt wurden. Die Einstellungen des Statussystems, d.h. die Maskenregister, werden nicht beeinflußt. Gelöscht werden:
- Standard Event Status Register (ESR)
- EVENt-Teil des QUEStionable-Registers
- EVENt-Teile des OPERation-Registers
- Error/Event-Queue

5.1.2 *ESE

Standard Event Status Enable setzt das Standard Event Status Enable Register auf den angegebenen Wert. Bit 5 des Status Bytes (ESB = Event Summary Bit) wird aus der „VERUNDUNG“ von ESR und ESE Register gebildet.
Die Abfrage *ESE? liefert den Inhalt des ESE-Registers in dezimaler Form zurück.

5.1.3 *ESR?

Standard Event Status Query liefert den Inhalt des Standard Event Status Registers in dezimaler Form zurück. Das ESR-Register und das ESB-Bit im Status Byte werden gelöscht.

5.1.4 *IDN?

Identification Query liefert die Geräteidentifikation als String zurück. Die Antwort lautet zum Beispiel

„TET, ARGOS 1000, 12345, 1.03“

TET:	Hersteller
ARGOS 1000:	Gerätebezeichnung
12345:	Seriennummer
1.03:	Firmware-Revisionsnummer

5.1.5 *IST?

Individual Status Query gibt den Inhalt des ist-Flags zurück (0 oder 1). Das ist-Flag entspricht dem Status-Bit, das bei einem Parallel Poll auf einer festgelegten Datenleitung übertragen wird.

5.1.6 *OPC

Operation Complete setzt das Bit 0 im Standard Event Status Register, wenn alle vorausgegangenen Befehle abgearbeitet wurden. Dieses Bit kann zum Erzeugen eines Service Request benutzt werden. `*OPC` sollte als letzter Befehl in einer terminierten Befehlsfolge benutzt werden.

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

Das `OPC`-Bit im Standard Event Status Register wird gesetzt, sobald die Ausgangsspannung stabil innerhalb der Toleranz an den Ausgangsklemmen anliegt (oder in die Strombegrenzung geht).

5.1.7 *OPC?

Operation Complete Query schreibt den Wert „`1`“ in den Ausgabepuffer, sobald alle vorausgegangegen Befehle abgearbeitet wurden. `*OPC?` sollte als letzter Befehl in einer terminierten Befehlsfolge benutzt werden.

5.1.8 *PRE

Parallel Poll Register Enable setzt das Parallel Poll Enable Register auf den angegebenen Wert. Dieses Register legt fest, welche Bits im Status Byte zum ist-Flag summiert werden. Das ist-Flag wird während einer Parallel-Poll-Abfrage gesendet.

5.1.9 *RST

Reset versetzt das Gerät in einen definierten Grundzustand. Für jede programmierbare Funktion ist dieser Grundzustand als *RST-Wert angegeben. *RST nimmt keine Änderungen an den Event- und Enable-Registern des Statussystems vor. Der Ausgabepuffer wird nicht gelöscht.
Spannung und Strombegrenzung werden auf 0 gesetzt, der Über-spannungsschutz auf 120% der Nennspannung. Der Ausgang wird eingeschalten (OUTPut = ON).

5.1.10 *SRE

Service Request Enable setzt den Wert des Service Request Enable Registers. Das SRE-Register legt fest, welche Bits im Status Byte einen Service Request verursachen dürfen.
Die Abfrage *SRE? Liefert den Inhalt des Service Request Enable Registers in dezimaler Form. Bit 6 ist dabei immer 0.

5.1.11 *STB?

Status Byte Query liest den Inhalt des Status Bytes in dezimaler Form aus. Bit 6 ist das MSS-Bit. Das MSS-Bit zeigt an, dass das Gerät Service angefordert hat, auch wenn das Gerät bereits gepolled und dabei das RQS-Bit gelöscht wurde.

5.1.12 *TST?

Self Test Query führt einen internen Selbsttest des Gerätes durch und gibt einen Fehlercode in dezimaler Form aus (0, „No Error“).

5.1.13 *WAI

Wait-to-Continue erlaubt die Abarbeitung der nachfolgenden Befehle erst, wenn die vorangegangenen Befehle abgearbeitet sind.

5.2 Spezielle RS232-Steuerbefehle

Bei GPIB geht das Gerät automatisch beim Empfang der ersten Nachricht in den Remote-Betrieb und kann bei Bedarf über den IEEE488.1 – Befehl <GTL> in den Local-Zustand gebracht werden.

Wenn das Netzteil über RS232 gesteuert wird, muss im Gegensatz zu GPIB eine Umschaltung von Local auf Remote und umgekehrt über einen expliziten Steuerbefehl erfolgen.

5.2.1 @REM

Mit dem Befehl @REM wird das Netzteil in den Zustand „Remote“ ver-setzt, sichtbar an der LED „REM“.

5.2.2 @LOC

Mit dem Befehl @LOC wird das Netzteil in den Zustand „Local“ ver-setzt.

Befehl	Parameter	Bemerkung
@REM		REMOTE-Umschaltung nur RS232 keine Abfrage
@LOC		LOCAL-Umschaltung nur RS232 keine Abfrage

5.3 MEASure – Subsystem

Befehl	Parameter	Bemerkung
MEASure **[:SCALar]** **:VOLTage[:DC]?** **:CURRent[:DC]?** **:AUXiliary[:DC]?**		 nur Abfrage nur Abfrage nur Abfrage

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

Dieser Befehl misst die Spannung in am Ausgang des Netzteils. Das Datenformat entspricht einer Gleitpunktzahl. Die Einheit ist Volt.

Beispiel: `'MEAS:VOLT?'`
Antwort z.B: `'23.507'`

Beachten: `'MEAS:VOLT?'` und `'VOLT?'` sind zwei unterschiedliche Befehle.
Während `'MEAS:VOLT?'` die Spannung tatsächlich misst, liefert `'VOLT?'` den zuletzt eingestellten Sollwert.

Eigenschaften: *RST-Wert: --
SCPI: konform

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

Dieser Befehl misst den Strom am Ausgang des Netzteils. Das Daten-format entspricht einer Gleitpunktzahl. Die Einheit ist Ampere.

Beispiel: `'MEAS:CURR?'` Antwort z.B: '56.466'
Beachten: `'MEAS:CURR?'` und `'CURR?'` sind zwei unterschiedliche Befehle. Während `'MEAS:CURR?'`den Strom tatsächlich misst, liefert `'CURR?'` den zuletzt eingestellten Sollwert
Eigenschaften: *RST-Wert: --
SCPI konform

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

Dieser Befehl liest den momentanen Wert des Überspannungs-schutzes. Das Datenformat entspricht einer Gleitpunktzahl. Die Einheit ist Volt.

Beispiel: `'MEAS:AUX?'` Antwort z.B: `'60.229'`
Eigenschaften: *RST-Wert: --
SCPI: gerätespezifisch

5.4 OUTPut – Subsystem

Befehl	Parameter	Bemerkung
OUTPut **[:STATe]**	<Boolean>	

5.4.1 OUTPut[:STATe] ON|OFF

Dieser Befehl schaltet den Ausgang ein oder aus.

Beispiel: `'OUTP OFF'`
`'OUTP?'` Antwort: `'0'`
Eigenschaften: *RST-Wert: OFF
SCPI: konform

5.5 SOURce – Subsystem

Befehl	Parameter	Bemerkung
[SOURce]		
:CURRent		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:VOLTage		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	<numeric_value>	
:PROTection		
:CLEar		keine Abfrage
[:LEVel]	<numeric_value>	
:TRIPped?		nur Abfrage

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

Sollwert für den Strom bzw. die Strombegrenzung in Ampere

Beispiele:	`'CURR 2.34'`	Sollwert = 2.34A
	`'CURR min'`	Sollwert = 0A
	`'CURR max'`	Sollwert = Nennstrom
	`'CURR 1500mA'`	Sollwert = 1.5A
	`'CURR?'`	Antwort: `'1.5'`
Eigenschaften:	*RST-Wert:	0 Ampere
	SCPI:	konform

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

Sollwert für die Spannung in Volt.

Beispiele:	`'VOLT 12.0V'`	Sollwert = 12V
	`'VOLT MIN'`	Sollwert = 0V
	`'VOLT MAX'`	Sollwert = Nennspannung
	`'VOLT 100'`	Sollwert = 100V
	`'VOLT?'`	Antwort: `'100'`
Eigenschaften:	*RST-Wert:	0 Volt
	SCPI:	konform

5.5.3 [SOURce:]VOLTage:PROTection:CLEar

Dieser Befehl setzt den Überspannungsschutz (OVP) zurück, wenn dieser ausgelöst hat.

Beispiel:	`'VOLT:PROT:CLE'`	
Eigenschaften:	*RST-Wert:	--
	SCPI:	konform

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

Dieser Befehl stellt die Spannung ein, bei welcher der Überspannungs-schutz (OVP) auslöst. Nach dem Einschalten und nach einem *RST ist dieser auf 120% der Nennspannung eingestellt.

Beispiele:	`'VOLT:PROT 48V'`	OVP = 48 Volt
	`'VOLT:PROT min'`	OVP = 0 Volt
	`'VOLT:PROT max'`	OVP = 120% Nennspg.
	`'VOLT:PROT DEF'`	OVP = 120% Nennspg.
	`'VOLT:PROT?'`	Antwort: `'48'`
Eigenschaften:	*RST-Wert:	120 % der Nennspannung
	SCPI:	konform

5.5.5 [SOURce:]VOLTage:PROTection:TRIPped?

Dieser Befehl fragt den Zustand des Überspannungsschutzes (OVP) ab. Hat der Überspannungsschutz ausgelöst, wird eine „1“ zurück-geliefert, ansonsten ein „0“.

Beispiel:	`'VOLT:PROT:TRIP?'`	Antwort: `'0'` oder `'1'`
Eigenschaften:	*RST-Wert:	--
	SCPI:	konform

5.6 STATus – Subsystem

Das Status-Subsystem enthält die Befehle zum Maskieren und Auslesen der OPERation- und QUEStionable-Register (siehe Kapitel 4.3). Das Statussystem wird nicht durch den Resetbefehl *RST beeinflußt. Anstelle dessen wird der Befehl *CLS (Clear Status) benutzt. *CLS löscht alle Event-Register und die Error/Event-Queue.

Befehl	Parameter	Bemerkung
STATus		
:OPERation		
:CONDition?		nur Abfrage
:ENABle	0 ... 65535	
[:EVENt?]		nur Abfrage
:PRESet		keine Abfrage
:QUEStionable		
:CONDition?		nur Abfrage
:ENABle	0 ... 65535	
[:EVENt?]		nur Abfrage

5.6.1 STATus:OPERation:CONDition?

Fragt den Inhalt des STATus:OPERation:CONDition-Registers ab. Der Inhalt des Registers bleibt erhalten. Er ist abhängig vom Hardwarezustand des Gerätes. Das Register kann durch eine Abfrage nur gelesen, jedoch nicht verändert werden. Die Bedeutung der einzelnen Bits des CONDition-Registers ist in Kapitel 4.3 dargestellt. Die Abfrage liefert einen dezimalen Zahlenwert im Bereich von 0 ... 65635, der dem Inhalt des 16-Bit-Registers entspricht.

Beispiel: `'STAT:OPER:COND?'` Antwort `'0'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.2 STATus:OPERation:ENABle 0 ... 65535

Dieser Befehl setzt die Bits des ENABle-Teils des STATus:OPERation-Registers. Damit wird bei auftretenden Events im STATus: OPER-ation-Teil das Summenbit „OPER“ gesetzt. Das Summenbit ist Teil des Status Bytes und kann zum Erzeugen eines Service Requests benutzt werden.

Beispiel: `'STAT:OPER:ENAB #H3039'`
`'STAT:OPER:ENAB?'` Antwort `'12345'`

Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.3 STATus:OPERation[:EVENt]?

Fragt den Inhalt des STATus:OPERation[:EVENt]-Registers ab. Die Abfrage liefert einen dezimalen Zahlenwert im Bereich von 0 ... 65635, der dem Inhalt des 16-Bit-Registers entspricht. Das EVENt-Register wird bei der Abfrage gelöscht.

Beispiel: `'STAT:OPER?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.4 STATus:PRESet

Dieser Befehl setzt die ENABle-Teile von STATus:OPERation und STATus:QUEStionable auf 0. Die Register Standard Event Enable (ESE) und Service Request Enable (SRE) werden nicht gelöscht. Sie müssen explizit mit den Anweisungen `'*ESE 0'` bzw. `' *SRE 0'` gelöscht werden.

Beispiel: `'STAT:PRES'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.5 STATus:QUEStionable:CONDition?

Fragt den Inhalt des STATus:QUEStionable:CONDition-Registers ab. Der Inhalt des Registers bleibt erhalten. Er ist abhängig vom Hardwarezustand des Gerätes. Das Register kann durch eine Abfrage nur gelesen, jedoch nicht verändert werden. Die Bedeutung der einzelnen Bits des CONDition-Registers ist in Kapitel 4.3 dargestellt. Die Abfrage liefert einen dezimalen Zahlenwert im Bereich von 0 ... 65635, der dem Inhalt des 16-Bit-Registers entspricht.

Beispiel: `'STAT:QUES:COND?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.6 STATus:QUEStionable:ENABle 0 ... 65535

Dieser Befehl setzt die Bits des Enable-Teils des STATus:QUES-tionable-Registers. Damit wird bei auftretenden Events im STATus: QUEStionable-Teil das Summenbit „QUES“ gesetzt. Das Summenbit ist Teil des Status Bytes und kann zum Erzeugen eines Service Requests benutzt werden.

Beispiel: `'STAT:QUES:ENAB 12345'`
`'STAT:QUES:ENAB?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.6.7 STATus:QUEStionable[:EVENt]?

Fragt den Inhalt des STATus:QUEStionable[:EVENt]-Registers ab. Die Abfrage liefert einen dezimalen Zahlenwert im Bereich von 0 ... 65635, der dem Inhalt des 16-Bit-Registers entspricht. Das EVENt-Register wird bei der Abfrage gelöscht.

Beispiel: `'STAT:QUES?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.7 SYSTem – Subsystem

Befehl	Parameter	Bemerkung
SYSTem		
:ERRor		
[:NEXT]?		nur Abfrage
:VERSion?		nur Abfrage

5.7.1 SYSTem:ERRor[:NEXT]?

Dieser Befehl fragt den ältesten Eintrag der Error Queue ab und löscht ihn dadurch. Positive Fehlernummern bezeichnen gerätespezifische Fehler, negative Fehlernummern von SCPI festgelegte Fehlermeldungen. Wenn die Error Queue leer ist, dann wird `'0, "No error"'` zurückgegeben, ansonsten eine entsprechende Fehlernummer.
Dieser Befehl kann solange ausgeführt werden, bis alle Fehler ausgelesen wurden und zuletzt eine `'0, "No error"'` zurück-gegeben wird.
Mögliche Fehlernummern sind in Kapitel 6 beschrieben.

Beispiel: `'SYST:ERR?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

5.7.2 SYSTem:VERSion?

Dieser Befehl fragt die SCPI-Versionsnummer ab, zu der das Gerät konform ist. Die Antwort wird im Format YYYY.V geliefert, wobei YYYY dem Jahr und V der Version entspricht.

Beispiel: `'SYST:VERS?'`
Eigenschaften: *RST-Wert: --
SCPI: konform

6 Fehler- und Eventnummern

6.1 SCPI-Fehler- und Eventnummern

Code	*Fehlertext*	*Beschreibung*
0	**No error**	Errorqueue enthält keine Einträge
-100	**Command Error**	Fehlerhafter Befehl
-102	**Syntax Error**	Befehl ist ungültig
-108	**Parameter not allowed**	Der Befehl enthält zu viele Parameter
-109	**Missing Parameter**	Der Befehl enthält zu wenig Parameter
-110	**Command Header Error**	Der Header des Befehls ist fehlerhaft
-115	**Unexpected Number of Parameters**	Der Befehl enthält zu viele Parameter
-200	**Execution Error**	Fehler bei der Ausführung eines Befehls
-220	**Parameter Error**	Der Befehl enthält einen fehlerhaften oder ungültigen Parameter
-222	**Data out of range**	Der Parameterwert liegt ausserhalb des vom Gerät erlaubten Bereichs.
-240	**Hardware Error**	Befehl kann wegen fehlender Hardware nicht ausgeführt werden
-300	**Device-specific error**	Gerätespezifischer Fehler
-350	**Queue Overflow**	Dieser Code wird bei einem auftretenden Fehler in die Errorqueue eingetragen, wenn diese voll ist. Die Errorqueue kann nur 5 Einträge aufnehmen.
-400	**Query Error**	Allgemeiner Fehler bei der Datenanforderung
-410	**Query Interrupted**	Abfrage wurde unterbrochen, weil z.B. das Gerät neue Daten empfängt, bevor die Antwort vollständig geschickt wurde.
-420	**Query Unterminated**	Das Gerät hat eine unvollständige Abfrage erhalten.
-430	**Query Deadlocked**	Eine Abfrage kann nicht verarbeitet werden, weil Ein- und Ausgabepuffer voll sind.
-500	**Power On**	Wird nach dem Einschalten in die Error-/Eventqueue geschrieben
-800	**Operation Complete**	Alle Befehle wurden ausgeführt (siehe Befehle *OPC und *OPC?)

6.2 Gerätespezifische Fehlermeldungen

Code	*Fehlertext*	*Beschreibung*
1xx	**internal Hardware Error**	Hardwarefehler
200	**UART Error**	Baudrate, Parity etc. fehlerhaft eingestellt
210	**DAC Error**	D/A-Wandler arbeitet fehlerhaft
220	**Flash Write Error**	Fehler beim Sichern eines Parameters
230	**Flash Erase Error**	Fehler beim Löschen des internen Flashspeichers
260	**Temperature Error**	Übertemperatur
270	**Overvoltage Protection Tripped**	Überspannungsschutz hat ausgelöst
280	**Watchdog Timer**	Watchdog Timer hat einen Reset verursacht

7 TET-Befehle

7.1 Hinweise zur Programmierung

Grundsätzlich ist zu beachten:

1. Um den TET-Befehlssatz zu benutzen, muss der Jumper J3 „SYNTAX“ gesetzt werden.
2. Wenn der abwärtskompatible TET-Befehlssatz bentutzt wird, ist das Gerät nicht konfom zum IEEE 488.2 – Standard.
   Insbesondere das Statusbyte und die Erzeugung eines Service Requests (SRQ) weichen vom Standard ab (siehe Kapitel 8.4).
   „Common Commands“ wie in Kapitel 5 beschrieben sind nicht verfügbar, bis auf die Befehle **IDN?* und **STB?*.
3. Jede Eingabe von Sollwerten (*V*, *C*, oder *L*) wird erst mit dem Zeichen *X* wirksam, z.B: *V30X*.
4. Der Überspannungsschutz muss immer um 10% oder 1 Volt höher als die gewünschte Ausgangsspannung sein.
5. Das Interface erwartet, daß alle Befehle in ASCII kodiert sind, akzeptiert jedoch große und kleine ASCII-Zeichen. Die Datenausgabe erfolgt in großen Zeichen.
6. Bei der Übertragung eines Datenblocks wird dessen Ende vom Talker mit einem Abschlusszeichen (z.B. *EOI*) versehen. Um eine Kommunikation zwischen Rechner und Interface zu ermöglichen muss das Interface und der Rechner auf das entsprechende Abschlusszeichen eingestellt werden (siehe Kapitel 2).
7. Im Gegensatz zum SCPI- bzw. IEEE 488.2 – Standard werden beim TET-Befehlssatz alle Befehle vom Interface beantwortet bzw. quittiert (siehe Befehlsliste).
8. Eine Verkettung von Befehlen ist möglich, indem die einzelnen Befehle durch ein Semikolon „;“ getrennt werden.
   Im zurückgelieferten String werden die Antworten auf die einzelnen Befehle durch ein Komma getrennt.
   Beispiel:
   Befehl: *f0; l120x; f1* Antwort: *F_0, >, F_1*

7.2 Sollwert-Programmierung

7.2.1 Eingabe-Modus

Die Sollwerte für Spannung (*V*), Strom (*C*) und Überspannungsschutz (*L*) können entweder in Prozent (bezogen auf den Nennwert) oder im Fließkommaformat eingegeben werden.

Nach einem Systemstart ist das Fließkommaformat eingestellt.

Prozentmodus wird mit dem Befehl „*F0*“ aktiviert.

„*F0*“: Prozentmodus

„*F1*“: Fließkommaformat

Zu beachten ist, dass sämtliche Sollwerteingaben erst mit dem Freigabebefehl „*X*“ wirksam werden.

Eine Möglichkeit, die Ausgangsspannung auf 0 Volt zu programmieren, ohne den Sollwert zu ändern, bietet der Befehl „*N*“.

Nach Eingabe des Zeichens „*X*“ wird der Befehl „*N*“ aufgehoben und die zuletzt mit „*V*“ programmierte Ausgangsspannung eingestellt.

7.2.2 Beispiele

V0C0L0X	- setzt Spannung, Strom und Überspannungsschutz auf 0
F1 *V10.25X* *C100X* *L1.2E2X*	- Floatmodus einstellen - Spannung auf 10,25 V einstellen - Strombegrenzung auf 100 A einstellen - Überspannungsschutz auf 120 V einstellen
F0 *V75.23X* *L120X* *N* *X*	- Prozentmodus einstellen - Spannung auf 75,23% der Nennspannung einstellen - Überspannungsscchutz auf 120% von Unenn setzen - Spannung auf 0 Volt einstellen, Sollwert beibehalten - Sollwert wieder freigeben

Tabelle 10: Beispiele für die Sollwertprogrammierung

7.3 Istwert-Messung

Mit den Befehlen M0 bis M7 wird der gewünschte Messmodus eingestellt. Mit dem Messmodus wird festgelegt, welcher Istwert als Antwort auf den Befehl „M“ zum Steuerrechner übertragen wird:

Messmodus	***Zu übertragender Istwert***
M0	Kein Messwert
M1	Spannung
M2	Strom
M3	Spannung und Strom (nach dem Einschalten voreingestellt)
M4	AUX (bei Atlas, Argos, Hellas, Hercules = OV-Limit)
M5	Spannung und AUX
M6	Strom und AUX
M7	Spannung, Strom und AUX
Mess	Die durch *M0-M7* eingestellten Istwerte werden kontinuierlich an der RS232-Schnittstelle ausgegeben.

Tabelle 11: Messmodi

Durch Senden des Zeichens „*M*“ werden die Istwerte zum Rechner übertragen.

Achtung: Die Istwerte werden immer als Absolutwerte übertragen, auch wenn mit „*F0*“ der Prozentmodus eingestellt ist.

Beispiel:

Befehl	***Antwort***
M3	*>*
M	*01_V:23.12340000_C:2.434500000*

Befehl	*Antwort*
M7	*>*
M	*01_V:23.12340000_C:2.434500000_X:48.0230000*

Tabelle 12: Beispiele für die Istwertmessung

7.4 Statusbyte

Das Status-Reporting ist bei Verwendung der alten Steuerbefehle (TET-Syntax) nicht konform zu IEEE488.2. Um Steuersoftware weiterhin zu unterstützen, die mit älteren Versionen der Option 34 programmiert wurde, wurde auf eine Anpassung an IEEE488.2 verzichtet.

Soll ein Steuerprogramm erzeugt werden, das ein IEEE488.2-konformes Status-Reporting besitzt, so ist der Umstieg auf die SCPI-Syntax notwendig.

Die Statusabfrage dient in erster Linie dazu, bei der Stromversorgung ein Ansprechen der Strombegrenzung (CC) und/oder des Überspannungsschutzes (OV) zu erkennen.

7.4.1 Aufbau des Statusbytes

Bit	*Name*	*Bedeutung*
0	OVM	Momentaner Zustand des Überspannungs-schutzes
1	CCM	Momentaner Zustand der Strombegrenzung
2	OVG	Gerasteter Zustand des Überspannungs-schutzes
3	CCG	Gerasteter Zustand der Strombegrenzung
4	0	
5	0	
6	SRQ	SRQ-Anforderung
7	ERR	Fehler ist aufgetreten -> siehe Errorqueue

7.4.2 Abfrage des Statusbytes

mit Befehl „W“:

Durch Eingabe des Zeichens „*W*“ sendet das Interface das Statusbyte in der Form „*01_W_xxxxxxxx*“.

Spricht z.B. momentan die Strombegrenzung des Gerätes an, so liefert der Befehl „*W*“ die Antwort „*01_W_00001010*“.

mit einem Serial Poll:

Die Stromversorgung wird mit dem Befehl „Q“ so konfiguriert, dass bei Auslösen des Überspannungsschutzes (OV) und/oder der Strom-begrenzung (CC) ein Serial Request (SRQ) erzeugt wird.
Der GPIB-Controller holt sich bei Auftreten eines SRQ von allen am Bus hängenden Geräten das Statusbyte ab (Serial Poll) und entscheidet anhand Bit 6 des jeweilgen Statusbyte, welches Gerät Service anfordert.

7.4.3 Statusbyte zurücksetzen

Das Erzeugen eines Service Request (SRQ) kann mit dem DIL-Schalter S1.7 unterdrückt werden (ON = SRQ erlaubt, OFF = SRQ unterdrückt).

Das Statusbyte wird mit dem Befehl *&0* bzw. *&1* zurückgesetzt. Einträge in der Fehlerqueue werden dabei gelöscht.

&0: Statusbyte zurücksetzen und weitere SRQs unterdrücken.
&1: Statusbyte zurücksetzen und weitere SRQs erlauben.

Die Befehle *&0* und *&1* überschreiben die Einstellung von Schalter S1.7. Wurde das Gerät mit DIL-Schalter S1.7 zum Beispiel so konfiguriert, dass keine SRQs erlaubt sind, dann kann mit dem Befehl *&1* dieses Verhalten geändert werden.

7.4.4 SRQ-Auslösung maskieren

Mit dem Befehl „Q“ kann das Erzeugen eines SRQ auf die Auslösung des Überspannungsschutzes (OV) und/oder der Strombegrenzung (CC) maskiert werden (siehe Tabelle 13).

Befehl	***Maskierung***
Q00	Kein SRQ
Q01	SRQ bei CC
Q10	SRQ bei OV
Q11	SRQ bei CC und OV

Tabelle 13: SRQ-Maskierung

7.5 Fehlerbehandlung

Auftretende Fehler werden in einer Error-Queue (FIFO) gespeichert. Die Queue enthält die letzten 5 aufgetretenen Fehler.

Sind bereits 5 Fehler aufgetreten und es tritt ein weiterer Fehler auf, so wird der letzte Fehler durch die Nummer -350, Queue Overflow, ersetzt.

Der Befehl „Y“ fragt den ältesten Eintrag der Error-Queue ab und löscht ihn dadurch.

Positive Fehlernummern bezeichnen gerätespezifische Fehler, negative Fehlernummern von SCPI festgelegte Fehler-meldungen. Wenn die Error Queue leer ist, dann wird eine 0, „No error“ zurückgegeben, ansonsten eine entsprechende Fehler-nummer.

Der Befehl „Y“ kann solange ausgeführt werden, bis alle Fehler ausgelesen wurden und zuletzt eine „0“ zurückgegeben wird.

Mögliche Fehlernummern sind in Kapitel 6 beschrieben.

7.6 Befehlsliste

Jeder Befehl wird vom Interface beantwortet, entweder mit den angefragten Daten, oder mit

„>“ : Quittierung, d.h. Befehl wurde ausgeführt

„!“ : Fehler ist aufgetreten

***IDN?**	Identification Query
Typ:	Abfrage
Verwendung:	Identifikation des Gerätes am IEEE488- Bus
Abfrage-Syntax:	*IDN?
Antwort:	String mit folgender Zusammensetzung: HERSTELLER, MODELL, SERIENNUM-MER, FIRMWARE-REVISIONSNUMMER

Beispiel: TET, ARGOS 2000, 0, 1.0

#	Identification Query – identisch mit *IDN?
Typ:	Abfrage
Verwendung:	Identifikation des Gerätes am IEEE488- Bus
Abfrage-Syntax:	#

B	Remote Control (nur RS232)
Typ:	Einstellung oder Abfrage
Verwendung:	Umschalten der Stromversorgung auf Fern-steuerung
Einstell-Syntax:	B1: Remote-Control B0: Manual-Control
Abfrage-Syntax:	B
Antwort:	B1: Remote-Control B0: Manual-Control
Default:	B0
Beschreibung:	Soll die Stromversorgung über RS232 ferngesteuert werden, dann muss das Gerät mit „B1“ in den Remote-Zustand gebracht werden. Bei Fernsteuerung über IEEE488 geschieht dies automatisch.

C	Current (Stromsollwert einstellen)
Typ:	Einstellung
Verwendung:	Ausgangsstrom programmieren
Einstell-Syntax:	C<Zahl>
Antwort:	> oder !
Default:	C0
Beschreibung:	Dieser Befehl stellt den Stromsollwert auf den gewünschten Wert ein. **Vorsicht:** Prozent-/Floatmodus beachten !
Beispiel: C50X	Floatmodus: Strombegrenzung auf 50 A schalten Prozentmodus: Strombegrenzung auf 50% vom Nennstrom setzen

---

F	Float- / Prozentmodus
Typ:	Einstellung oder Abfrage
Verwendung:	Umschalten zwischen Float- und Prozent-modus
Einstell-Syntax:	F0: Prozentmodus F1: Floatmodus
Abfrage-Syntax	F
Antwort:	F_0: Prozentmodus F_1: Floatmodus
Default:	F1

Beschreibung: Nach einem Systemstart erwartet das Interface die Eingabe der Sollwerte V, C und L im Floatmodus. Bei Bedarf kann mit „F0“ in den Prozentmodus umgeschaltet werden.

Beispiel:

F0	Überspannungsschutz auf 120% der Nenn-spannung einstellen.
L120X	Anschliessend wieder in Floatmodus wechseln.
F1	

I	Digitale Eingänge lesen
Typ:	Abfrage
Verwendung:	Lesen der digitalen Eingänge DIN P1, DIN P2, DIN P3
Abfrage-Syntax	I
Antwort:	I_xyz mit: x = DIN P1 : Überspannungsschutz (OV) y = DIN P2 : Strombegrenzung (CC) z = DIN P3 : Übertemperatur (OT)
Default:	--

Beispiel:

I	Antwort: I_010

L	LIMIT (OV, Überspannungsschutz)
Typ:	Einstellung
Verwendung:	Überspannungsschutz programmieren
Einstell-Syntax:	L<Zahl>
Antwort:	> oder !
Default:	L = 120% Unenn

Beschreibung: Dieser Befehl stellt den Überspannungs-schutz auf den gewünschten Wert ein.
Vorsicht: Prozent-/Floatmodus beachten !

Beispiel: L30X Floatmodus: Überspannungsschutz auf 30 V schalten
Prozentmodus: Überspannungsschutz auf 30% Unenn schalten

M	Messen der Istwerte
Typ:	Einstellung oder Abfrage
Verwendung:	Messung der Istwerte von Spannung, Strom und AUX
Einstell-Syntax:	M0: kein Messwert ausgewählt M1: Messung Spannung M2: Messung Strom M3: Messung Spannung und Strom M4: Messung AUX M5: Messung Spannung und AUX M6: Messung Strom und AUX M7: Messung Spannung, Strom und AUX
Einstell-Antwort:	> oder !
Abfrage-Syntax:	M
Abfrage-Antwort:	M0: > M1: 01_V:<Zahl> M2: 01_C:<Zahl> M3: 01_V:<Zahl>_C:<Zahl> M4: 01_X:<Zahl> M5: 01_V:<Zahl>_X:<Zahl> M6: 01_C:<Zahl>_X:<Zahl> M7: 01_V:<Zahl>_C:<Zahl>_X:<Zahl>
Default:	M3
Beschreibung:	Dieser Befehl erlaubt die Abfrage der Istwerte von Spannung, Strom und AUX. Der Meßmodus kann jederzeit mit M0-M7 geändert werden. Im Antwortstring kenn-zeichnen die beiden ersten Ziffern die Kanalnummer, die bei Stromversorgungen mit einem Ausgang stets „01“ ist. <Zahl> entspricht dem jeweiligen absoluten Istwert und wird als Gleitkommazahl ohne Exponent gesendet.
Beispiel:	Antwort:
M6	>
M	01_C:3.12230000_X:47.34560000

Mess	Messwerte auf RS232 ausgeben
Typ:	Einstellung
Verwendung:	Kontinuierliche Anzeige der Messwerte
Einstell-Syntax:	Mess
Antwort:	> oder !
Beschreibung:	Die Messwerte werden kontinuierlich auf der RS232-Schnittstelle ausgegeben. Die Form der Ausgabe wird mit dem Befehl M0-M7 festgelegt.
Beispiel:	Ausgabe:
M3	>
Mess	>
	ständige Ausgabe auf RS232:
	01_V:9.22530000_C:62.12840000

N	NULL
Typ:	Einstellung
Verwendung:	Nullsetzen der Ausgangsspannung
Einstell-Syntax:	N
Antwort:	> oder !
Beschreibung:	Mit diesem Befehl wird die Ausgangs-spannung auf 0 Volt gesetzt, ohne den Sollwert zu verändern.

Beispiel:

V17.875X	Ausgangsspannung = 17.875 Volt
N	Ausgangsspannung = 0 Volt
X	Ausgangsspannung = 17.875 Volt

O	Digitale Ausgänge setzen / abfragen
Typ:	Einstellung oder Abfrage
Verwendung:	Digitale Ausgänge setzen / abfragen x: DOUT1 – Inverter EIN(0)/AUS(1)-Schalten y: DOUT2 – OVP rücksetzen z: DOUT3
Einstell-Syntax:	Oxyz
Antwort:	O_xyz
Abfrage-Syntax:	O
Antwort:	O_xyz

Beispiel:

O111	Antwort: O_111 - Alle Ausgänge wurden auf HIGH gesetzt
O	Antwort: O_111

Beispiel: Überspannungsschutz (OVP) zurücksetzen

L20X	>	// OVP auf 20 Volt setzen
V30X	>	// Ausgang auf 30Volt
I	I_100	// OVP hat ausgelöst
L48X	>	// OVP auf 48 Volt setzen
O010;O000	O_010,O_000	// OVP rücksetzen
I	I_000	// OVP ist rückgesetzt

P	Nennspannung / Nennstrom
Typ:	Einstellung oder Abfrage
Verwendung:	Abfragen von Nennspannung und Nenn-strom des Gerätes x: Nennspannung in Volt y: Nennstrom in Ampere
Einstell-Syntax:	Px y Beachte: Nennspannung und Nennstrom werden im Flashspeicher des Gerätes nichtflüchtig als Parameter abgelegt. Um ein versehentliches Überschreiben zu vermeiden, muss vor der Einstellung mit dem Befehl „SERVICE“ der Schreibschutz aufgehoben werden.
Antwort:	P_x_y
Abfrage-Syntax:	P
Antwort:	P_x_y
Beispiel:	
P	Antwort: *P40.000000_25.000000* - Nennwerte: 40V/25A
SERVICE P100 10	Antwort: > Antwort: *P100.00000_10.000000* - Gerät wurde umprogrammiert auf die Nennwerte 100V/10A

Q	SRQ-Maskierung
Typ:	Einstellung oder Abfrage
Verwendung:	Maskierung der SRQ-Generierung auf OV und/oder CC
Einstell-Syntax:	Q00: kein SRQ Q01: SRQ nur bei CC Q10: SRQ nur bei OV Q11: SRQ bei CC und OV
Abfrage-Syntax	Q
Antwort:	01_Q_00 01_Q_01 01_Q_10 01_Q_11
Default:	Q00
Beschreibung:	Mit diesem Befehl kann die SRQ-Anfor-derung auf die Ereigniss OV und/oder CC maskiert werden. Im Antwortstring kenn-zeichnen die beiden ersten Ziffern die Kanalnummer, die bei Stromversorgungen mit einem Ausgang stets „01“ ist.
Beispiel: &1 Q10	Antwort: &_1 // Statusbyte gelöscht, SRQ erlaubt 01_Q_10 // SRQ bei Überspannung

V	Voltage
Typ:	Einstellung
Verwendung:	Ausgangsspannung einstellen
Einstell-Syntax:	V<Zahl>
Antwort:	> oder !
Default:	V0

Beschreibung: Dieser Befehl stellt den Spannungssollwert auf den gewünschten Wert ein. **Vorsicht:** Prozent-/Floatmodus beachten !

Beispiel:
V30.83X

Floatmodus: Überspannungsschutz auf 30.83 V schalten
Prozentmodus: Überspannungsschutz auf 30.83% Unenn schalten

W	Statusbyte
Typ:	Abfrage
Verwendung:	Statusbyte lesen
Abfrage-Syntax:	W
Antwort:	01_W_<Bit 7..0>

Beschreibung: Dieser Befehl erlaubt die Abfrage des Statusbytes. m Antwortstring kennzeichnen die beiden ersten Ziffern die Kanalnummer, die bei Stromversorgungen mit einem Ausgang stets „01“ ist.

Beispiel:	Antwort:
V33.70X	>
W	01_W_00001010

Das heisst, die Stromversorgung wird mo-mentan an der Stromgrenze betrieben.

X	EXECUTE
Typ:	Einstellung
Verwendung:	Sollwerte freigeben
Einstell-Syntax	X
Antwort:	> oder !
Default:	V0

Beschreibung: Erst durch diesen Befehl wird die Stromversorgung auf die programmierten Sollwerte eingestellt.
Der Befehl „N“ wird durch „X“ aufgehoben.

Y	Fehlerabfrage
Typ:	Abfrage
Verwendung:	Fehlercode aus der Errorqueue auslesen
Abfrage-Syntax:	Y
Antwort:	<Fehlercode, Text>
Beschreibung:	Mit diesem Befehl wird der zuerst aufge-tretene Fehler ausgelesen. Wenn kein Fehler aufgetreten ist, wird „0, No error“ zurückgegeben (siehe Kapitel 8.5).

Beispiel:	Antwort:
Y	-102 // erster Fehler: Syntax-Error
Y	-222 // zweiter Fehler: Data out of Range
Y	0 // keine Fehler mehr vorhanden

&	Statusbyte, SRQ CLEAR
Typ:	Einstellung oder Abfrage
Verwendung:	Erlaubt oder verbietet die SRQ-Erzeugung. Statusbyte wird zurückgesetzt.
Einstell-Syntax:	&0: Statusbyte zurücksetzen und weitere SRQs unterdrücken. &1: Statusbyte zurücksetzen und weitere SRQs erlauben.
Abfrage-Syntax	&
Antwort:	&_0: SRQs werden unterdrückt &_1: SRQs sind erlaubt
Default:	&1
Beschreibung:	Mit diesem Befehl kann eine SRQ-Anforder-ung durch die Stromversorgung global erlaubt (&1) oder verboten (&0) werden. &0 und &1 setzen das Statusbyte zurück.

Beispiel:	Antwort:
W	01_W_00001100
&1	&_1 // Statusbyte gelöscht, SRQ erlaubt
W	01_W_00000000