Elektro-Automatik

PROGRAMMING GUIDE

MODBUS & SCPI

For 10000 and 20000 series

Attention! This document is only valid for the device series listed in section *1.1.2* and also only valid from the firmware versions listed there.

35400100_MA-Programming-10k-20k-EN_01_ia

TABLE OF CONTENTS

1. General

1.1 About this document

1.1.1 Copyright

Reprinting, copying, also partially, and usage for other purposes as foreseen of this manual are forbidden and breach may lead to legal process.

1.1.2 Validity

This document is valid for the below listed equipment including derived variants. This programming guide is connected to the firmware version of the communication unit KE. Older KE firmwares can thus be partially incompatible with this document. **The abbreviation** from the table below is later used as reference to determine whether a specific command or feature is supported by your device.

Series	Firmware from	Abbreviation
BT 20000	KE: 1.0.0	**BT**
All 10000 series together	KE: 3.02	**10K**
All 20000 series together	KE: 1.0.0	**20K**
PSI 10000	KE: 3.02	**PSI1**
PS 10000 / PU 10000	KE: 3.02	**PS1**
PSB 10000 / PUB 10000	KE: 3.02	**PSB1**
PSBE 10000	KE: 3.02	**PSBE1**
ELR 10000 / PUL 10000	KE: 3.02	**ELR1**

1.2 Explanation of symbols

Warning and safety notices as well as general notices in this document are shown in a box with a symbol as follows:

⚠	Symbol for general safety notices (instructions and damage protection bans)
❗	*Symbol for general notices*

1.3 Warranty

The manufacturer guarantees the functional competence of the applied technology and the stated performance parameters. The warranty period begins with the delivery of equipment free from defects.

Terms of guarantee are included in the general terms and conditions of the manufacturer.

1.4 Limitation of liability

All statements and instructions in this manual are based on current norms and regulations, up-to-date technology and our long term knowledge and experience. The manufacturer accepts no liability for losses due to:

- Usage for purposes other than designed
- Use by untrained personnel
- Modification of devices by the customer
- Technical changes of devices
- Use of unauthorized spare parts

2. First steps

2.1 Anybus modules

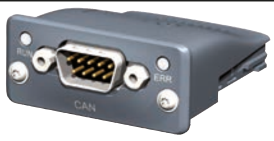
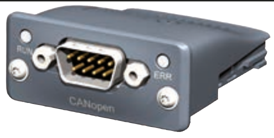
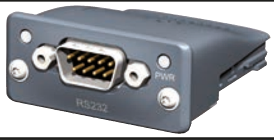
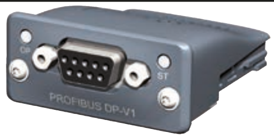
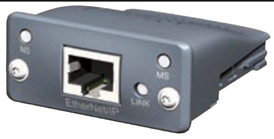
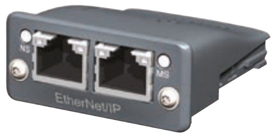
Besides built-in interfaces (USB, analog) a device might be equipped with, some device series support to install a further, digital interface. It comes in form of a small module, can be bought at a later date and retrofitted on location. The device would detect and configure the module automatically.

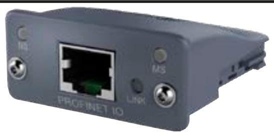
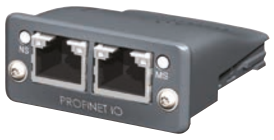
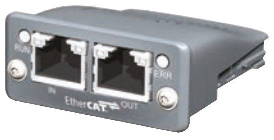
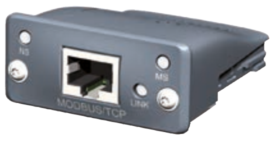
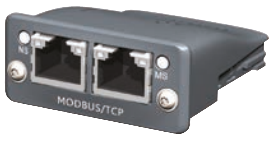
These device series support the in *2.1.1* listed Anybus modules (date: 02-27-2024):

- all 10000 series

It might be required to install a firmware update for the "KE" component in your device in case the device won't support a specific interface module after installation.

2.1.1 Module types

Module	Type / Connectors	LED indication		Front view
IF-AB-CAN	CAN 2.0B, 1x Sub-D 9pole, male	RUN	Indicates data traffic (green, flickering)	
		ERR	Will be on (green) while a communication error is present.	
IF-AB-CANO	CANopen, 1x Sub-D 9pole, male	RUN	Indicates status with flash sequences according to DR303-3 (CiA)	
		ERR	Indicates status with flash sequences according to DR303-3 (CiA)	
IF-AB-RS232	RS 232, 1x Sub-D 9pole, male, for null modem cable	PWR	Module is powered	
IF-AB-PBUS	Profibus DP-V1 Slave, 1x Sub-D 9pole, female	OP	Operation mode: on (green) = Connection established flashing (green) = Ready flashing (red, 1x) = Parameter error flashing (red, 2x) = Profibus error	
		ST	Status off = Not initialised on (green) = Initialised flashing (green) = Extended diagnosis on (red) = Exception error	
IF-AB-ETH1P	Ethernet, 1x RJ45	NS	Network status: flashing (green) = default, can be ignored on (red) = Double IP, fatal error flashing (red) = Connection time-out	
		MS	Module status: flashing (green) = default, can be ignored on (red) = Exception error flashing (red) = Recoverable error	
IF-AB-ETH2P	Ethernet, 2x RJ45	LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	

Module	Type / Connectors	LED indication		Front view
IF-AB-PNET1P	ProfiNET IO, 1x RJ45	NS	Network status: on (green) = Online with controller in RUN flashing (green) = Controller in STOP	
		MS	Module status: on (green) = Everything OK on (red) = Exception error flashing (red, 1x) = Config error flashing (red, 2x) = IP address not set flashing (red, 3x) = Station name not set flashing (red, 4x) = Internal error	
IF-AB-PNET2P	ProfiNET IO, 2x RJ45			
		LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	
IF-AB-ECT	EtherCAT Slave, 2x RJ45	RUN	Indicates status with flash sequences according to DR303-3 (CiA)	
		ERR	Indicates status with flash sequences according to DR303-3 (CiA)	
IF-AB-MBUS1P	ModBus TCP, 1x RJ45	NS	Network status: on (green) = Module active flashing (green) = Module waiting for connection on (red) = Double IP or fatal error flashing (red) = Process time-out	
		MS	Module status: on (green) = Everything OK on (red) = Primary error flashing (red) = Secondary error	
IF-AB-MBUS2P	ModBus TCP, 2x RJ45			
		LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	

2.1.2 Before you begin

If you plan to integrate a device into an existing network or field bus with any of these interfaces installed, note the following:

- All modules, but especially the Ethernet types which provide a web site, require a certain startup time each time the device is powered, which will delay their network readiness. Usually, the interface module is ready for communication as soon as the device is ready for operation.
- The readiness for operation may be indicated by the modules (with one of the LEDs) before the required startup time has run out. If one would try to contact an Ethernet module in order to access the website, the website might not be loaded completely or the browser might stop with a time-out error.

2.1.3 Installation of an Anybus module

The installation itself is described in the user manual of your device, as well as the required setup. Further information about installation and connection to field buses and networks can be found in publicly available documentation and similar sources.

The CANopen module IF-AB-CANO doesn't feature an internal termination resistor. Thus the required of having a bus termination resistor installed is fulfilled by the user according to the CAN bus requirements.

2.1.4 Network with linear topology

The Ethernet based modules for standard LAN, ModBus TCP and Profinet/IO are also available in a version with two ports. These provide the possibility to connect multiple devices in a linear topology and even to build a ring (DLR, device level ring) for extended safety against interruption. External switches can be spared and the many long network cables, like when having a star-shaped topology, can be reduced to a minimum.

The EtherCAT module, however, has two ports by default and always builds a ring because of the standard setup within EtherCAT systems. It's also Ethernet based, but can't be considered as LAN port.

2.2 Network access via HTTP

The Ethernet based modules, like for standard LAN, ModBus TCP or Profinet, and the integrated LAN port of some series offer a **website**. It's accessible with common browsers (Firefox, Chrome, Safari) by simply entering the IP address or the host name which has been assigned to the device. Accessing the website via the host name (default: Client) is only possible if the network runs a DNS or, when using direct connection, the PC runs a DNS and the domain/host name is already registered.

The default IP is **198.168.0.2**. All network parameters for the device/network interface can be changed or reset to defaults in the setup menu of the device (where featured).

The currently active IP address, along with other network related parameters like gateway, DNS address, subnet mask and MAC address, can also be read from an overview in the setup menu of devices where the series features an on-screen setup menu.

The website gives the user full control over the device by manually typing SCPI commands. It primarily serves for simple testing purposes. In case you want to continuously control a device or at least monitor it, please continue reading in section *"5. SCPI protocol"*.

The website, precisely the second page CONFIGURATION, allows for setting up network specific parameters, like when doing it in the device setup menu, and to write them to the device remotely, requiring prior activation of remote control by command SYST:LOCK ON.

2.3 Network access via TCP

All Anybus network modules, as well as the integrated LAN port of select series offer standard TCP access via the default port **5025** (user-selectable, see device operating guide for setup menu or similar). TCP data transfer is used for the external communication via **ModBus RTU** or **SCPI**. For **ModBus TCP** protocol, port **502** is reserved and exclusive.

The default port and other network related parameters can either be adjusted in the device's setup menu (if featured) or from outside via USB or on the website (see *2.2*).

TCP/IP socket connection (IP:port) is intended for normal remote control access to the device when using an Ethernet interface.

The TCP connection can be automatically disconnected by the device after a certain time without data transmission has elapsed. This is due to an adjustable timeout (default: 5 s). There is also another related option called "TCP keep-alive" which, if activated, deactivates the timeout temporarily, unless "TCP keep-alive" isn't functioning within the network. From a specific firmware version on (see firmware history of your particular device) it also supports to permanently turn off the connection timeout.

3. Introduction

3.1 Access to the device

After connecting the device via a digital interface to a PC, it's usually ready for access. Access is possible in several ways:

- Via a control and monitoring software supplied by the device manufacturer
- Via LabView VIs, supplied by the device manufacturer
- Via a custom programmed application which is usually created by the user
- Via other software, such as a terminal program that can send text messages (SCPI) or hexadecimal bytes (ModBus)
- Via internationally standardized software for CAN, CANopen, Profibus or EtherCAT etc.

3.2 Control locations

Control locations are those locations from where a device is accessed. There are basically two: direct access (manual control) and external access (remote control). The user can switch between control locations just as the situation requires.

Following control locations are defined for a device:

Control location as displayed on the device	Description
Nothing or **Remote: None** or LED "Remote" off	If there is no specific control location indicated, the device is free for external access and all interfaces for remote control are enabled
Remote: Analog, Remote: USB etc. LED "Remote" on	Remote control is active
Local	Remote control is blocked, the device can only be controlled manually

Digital remote control always requires to be activated by a specific command sent to the device. It can't be activated on the device, only allowed or forbidden, and doesn't automatically activate when sending any first command.

Certain device series offer a feature to interrupt or to block remote control of the device completely. This is a setting named **Allow remote control** or similar. In blocked state, the device will indicate **Local** in the display. Activating this block may be required in an emergency situation where a software permanently controls the device from remote and thus may prevent user interaction. With this, the user can temporarily interrupt remote control in order to access the device for switching the DC input/output off or changing a setting.

Activating **Local** condition will cause the following:

- In case remote control via one of the digital interfaces is currently active (**Remote: Ethernet** etc.), remote control will be deactivated and can be activated again later, once the **Local** condition has been deactivated again.
- In case that remote control via analog interface is currently active (**Remote: Analog**), then it's only interrupted as long as condition **Local** is active and returns automatically as soon as **Local** is deactivated, because the analog interface has a steady signal on pin REMOTE, which permanently defines “remote on” unless the plug on the analog interface has been removed.

3.3 Message timing and command execution time

The timing of communication, more precisely the control over the chronological run of two subsequent messages, isn't done by the device and lies in the responsibility of the user.

Rule of thumb: the device can't process incoming messages as fast they can be technically transferred by the hardware of the used interface and its specifications. Therefore it's important to time communication and wait a certain time before the next command is sent, no matter what interface is used. This doesn’t include protocol related data traffic, like it occurs for example between a Profibus slave and its Profibus master, because this traffic is handled by the interface itself, not by the device.

The technical specifications sheet of a device or entire series may, however, state a minimal communication interval which would then override the herein given value.

It also applies:

- Queries to the devices, i.e. commands that read something, are executed faster and may be sent more often and in shorter intervals
- Write commands, i.e. commands that set a value or a status, are not immediately executed when received and the delay before execution varies

3.3.1 Execution time when writing

Due to different internal design of the series, different types of microcontrollers, which control the hardware and besides do communication with the PC, and also different firmwares the time for processing a set value or status command varies. It means, there is no fixed time between setting a value and the corresponding reaction on the DC output/input, only a typical one.

3.3.2 Response time when reading

Reading something from a device is usually responded immediately, with a certain response time. There are generally two methods of communicating with a port:

1) Open port -> write query to port -> read response from port -> close port

2) Open port -> write query to port -> read response from port -> repeat write/read x times -> close port

Both methods have advantages and disadvantages. The primary advantage of method 2 over method 1 is that writing and reading can result in an even faster response time. The primary advantage of method 1 over method 2 is that closing the port also closes the connection which can make communication more stable, especially if the time between two write-read cycles is very long. With Ethernet based connections, however, opening and closing the sockets can have a unwanted side effect of too many local ports being occupied at the same time.

The values in the table below have been acquired using method 2.

The response times, as listed below, include the transmission of any data between the device and the controlling unit. For interfaces with a variable baud rate (CAN, CANopen) they are only valid for the highest baud rate setting.

		Typical response times (RS232 excluded)		
Series	**Protocol**	**USB**	**Ethernet based**	**CAN/CANopen Profibus**
BT 20000 ELR 10000 PSB 10000 / PSBE 10000 PS 10000 / PSI 10000	SCPI	<5 ms	<5 ms	-
	ModBus [1]	<5 ms	<5 ms	<3 ms

(1 Includes derived formats with CAN, CANopen, Profinet etc.

3.3.3 Timing of messages

The minimum time between two messages, as listed below, primarily depends on the typical response time as listed in *3.3.2*, because a response takes more time. With field buses, such as Profinet or EtherCAT, the cycle time for the bus master might be set to 1 ms, but when the below given value is bigger than that, let's say 5 ms, it means that only every 5 cycles a new values is allowed to be stored in the cyclic data object.

Series	**Minimum time between two messages**	**Recommended time between two messages**
ELR 10000 PSB 10000 / PSBE 10000 PS 10000 / PSI 10000	USB / CAN / CANopen: 5 ms Ethernet: 5 ms EtherCAT: 5 ms Profibus / Profinet: 5 ms	USB / CAN / CANopen: 10-15 ms Ethernet: 10 ms EtherCAT: 10 ms Profibus / Profinet: 10 ms
BT 20000	USB / CAN: Ethernet: EtherCAT: 1 ms	USB / CAN: Ethernet:

3.4 Overview of the communication protocols

All below listed device series support two protocols: **ModBus** and **SCPI**. Basically, both can be used via RS232, USB and some of the Ethernet based interfaces. Exceptions are those related to dedicated standards, such as CAN, CANopen, Profibus, Profinet or EtherCAT.

When using an Ethernet port with ModBus protocol it requires an additional distinction between ModBus RTU and ModBus TCP. Over Ethernet, the ModBus TCP reserved port 502 only supports ModBus TCP frames, while all other ports would only support frames with a ModBus RTU ("ModBus RTU over Ethernet") or SCPI message.

3.4.1 Which device series supports what protocol(s)?

	ModBus		**SCPI**
Series	**RTU**	**TCP**	
All 10000s series	✓	✓ (1	✓
All 20000s series	✓	✓	✓

(1 Only via installed interface module IF-AB-MBUS1P or IF-AB-MBUS2P

Special fact: **SCPI** and **ModBus RTU** can be used via USB or Ethernet arbitrarily. The device distinguishes them by the first byte of a message, **which has to be 0x00 or 0x01 for ModBus RTU with a 10000 series device and 0x00-0x03 with a 20000 series device**.

Depending on the selection of the interface and the protocol you are going to use, a different part of this documentation will be relevant. Users with a device supporting the optional interfaces modules (see section *2.1*) have a wider selection. The table below lists which **Anybus** module supports what protocol:

Interface type	**ModBus?**	**SCPI?**	**Other protocol**
CAN	no	no	*yes (see "8. CAN / CAN FD")*
CANopen	no	no	CANopen (see *"7. CANopen"*)
RS232	yes (see *"4. The ModBus protocol"*)	yes (see *"5. SCPI protocol"*)	no
Profibus	no	no	Profibus (see *"6. Profibus & Profinet"*)
Ethernet	yes, ModBus RTU (see *"4. The ModBus protocol"*)	yes (see *"5. SCPI protocol"*)	no
ProfiNet	no	no	Profinet (see *"6. Profibus & Profinet"*)
ModBus TCP	yes, ModBus TCP (see *"4.5.5. ModBus TCP"*)	yes (see *"5. SCPI protocol"*)	ModBus RTU (see *"4. The ModBus protocol"*)
EtherCAT	no	no	EtherCAT (see *"9. EtherCAT"*)

With the **20000 series**, more interfaces are already built in by default. These can cover several protocols per port. With date 10/2023, following protocol support is included or feasible:

Port	**Name**	**Already supported protocol**	**Available upon request**
D-Sub 9-polig	**CAN**	CAN 2.0 A, CAN 2.0B, CAN FD	CANopen
RJ45	**LAN**	SCPI, ModBus RTU, ModBus TCP	ProfiNet
RJ45	**EtherCAT**	EtherCAT	
USB-B	**USB**	SCPI, ModBus RTU	

3.5 Special characteristics of remote control

When using remote control via digital interface, some things have to be taken into account:

- Configuration or control of the function generator (where available) requires a certain procedure. This is described in *4.10.8* for ModBus or *5.4.12* for SCPI. The described procedure for ModBus basically also applies to the use of any other interface like CAN, CANopen, EtherCAT etc.
- Some ModBus registers or SCPI commands are intended for the setup of the device exactly like when doing it manually in the device setup menu (where featured). Those registers/commands are not particularly grouped or marked with colors and should only be used to switch between configurations of the device.
- The adjustment limits („Limits“, where available, see device manual), as adjustable in the device’s setup menu, limit related set values from every control location, i.e. also in remote control via digital interface. This can lead to unexpected communication errors coming from the device when a value is too high/low. With SCPI language being used, these errors are not even returned automatically, because it's typical with SCPI to receive error messages only upon request. In order to be sure whether a set value has been accepted by the device or not, reading the set value is the only way.

3.6 Fragmented messages on serial transmissions

With RS232 (where supported), Ethernet or even with USB it's possible, that the device receives fragmented messages. It means, a command is received in pieces together with a certain time gap and then interpreted by the device as multiple, but single and corrupt commands. Primarily SCPI commands are affected, because they are strings consisting of ASCII characters and don't have a checksum. Those strings could be sent by a serial interface character by character or as one single block, depending on the situation on the control side (PC). If a certain timeout elapses between to consecutive bytes, the message is considered as "completely received" by the device, due to the lack of a termination character, which isn't generally required for ModBus or SCPI.

There is a variable USB timeout implemented which can be configure manually on the device (except for PS/PSI 5000) or via remote control, for example via SCPI (see section *5.4.11*). If the communication between PC and device has a lot of communication errors due to possibly fragmented messages, the timeout should be increased step by step to eliminate the problem. It's advised to keep the timeout setting as low as possible, because at the end of every message the timeout has to elapse before the device can process the command.

When using SCPI, sending an additional termination character (typical LF, CR, or CRLF), which isn't always required but accepted, will terminate the timeout immediately and let the device consider the message as completely received.

3.7 Connection timeout

Socket connection to devices which support an Ethernet port have a connection timeout. This variable and user-adjustable timeout (see user manual of the device) closes the socket connection automatically on the device side if there was no communication going on between the device and a controlling unit (PC, PLC etc.) for the adjusted time. After the socket has been closed, connection can be established again anytime. The timeout becomes automatically ineffective, if the so-called "TCP keep-alive" (available since a specific KE firmware version) is activated and supported in the network. Since a specific firmware version (see firmware history) it's even possible to completely turn off that timeout.

3.8 Effective resolution when programming

All values related to voltage, current, power and resistance, that can be transferred to the device and which are forwarded via the power stages to the DC input/output of the device, have the same defined programmable resolution and also an effective resolution. While the **programmable resolution** of a value is the same for every series (0-100 % = 0 - 0xCCCC = **0 - 52428**), even when using SCPI, the **effective resolution** depends on the ADC/DAC used in the hardware and isn't the same in all series. See the table below.

The effective resolution determines the achievable step width on the DC output/input. It calculates as *step width = rated value ÷ effective resolution*. E. g., for the power supply PSI 9080-510 3U the step width of voltage would then be 80 V / **26214** = approx. 3 mV. For the current it would be 510 A / **26214** = approx. 19 mA.

However, tolerances add to the result when setting a value. The PSI 10080-510 3U from the example above has a voltage tolerance of max. 0.05%, as stated in the user manual. This is up to 40 mV. When setting, for example, 24 V the true output voltage is allowed to be within a range of 23.96 V to 24.04 V. If you would measure the actual output voltage with an external multimeter and it would read 24.03 V and you would want to it have closer to the desired 24 V, the software could adjust the set value in steps of approx. 3 mV to further approach the actual value to the set value.

Series	ADC/DAC (1	Effective resolution
All 10000 series	16 Bit	26214
All 20000 series	16 Bit	26214

3.9 Minimum frequency slope (arbitrary function generator)

The function generator featured in the device series 10000 and 20000, which can also generate DC sine waves, has another limit when working with a frequency sweep, i. e. when the start and end frequency of the desired wave aren't equal. The minimum required slope Δf/s (delta frequency per second) is 9.3, as also mentioned in the device user manual in the section for the arbitrary generator, and can't be bypassed. When writing sequence point data which would result in a wrong minimum of that slope, the device would either directly return an error when using ModBus or, when using SCPI, upon request. The SCPI error would be -222.

When, for instance, wanting to achieve a sweep from 1 Hz to 10 Hz, the question is: How much time does the generator need until the sine wave reaches 10 Hz? This time is calculated by the control software. The angle at the end of the resulting wave might then not necessarily be the same as at the start. Formula:

ModBus: t (µs) = |end frequency - start frequency| / 9.3 * 1000000 -> because the time value here is given in µs

SCPI: t (s) = |end frequency - start frequency| / 9.3 -> because the time values here is given in seconds

For a wave from 1 Hz to 10 Hz, the time value would be calculated as 967,741 µs or 0.9677 s. When setting this exact value, running the generator and recording the entire wave on an oscilloscope, the last full sine wave will have a period that equals 10 Hz.

3.10 Special situations

When remotely controlling and supervising a device or a system of devices, unexpected and special situations may occur which need extra handling.

3.10.1 Alarm "Power fail"

The power fail alarm (short PF, see user manual of the device for more details) is one of a few device alarms which shut down the DC output/input. In this case usually due to a short-term power outage or an AC supply fluctuation. After the PF alarm has gone, the device and thus the control unit cannot immediately continue to operate as before, the control unit needs to wait some extra time. This waiting time isn't equal with all series and there is also no signal for when the device would be ready to continue. The waiting time also only starts with the PF alarm being gone already.

All devices from the series 10000 and 20000 have a PF alarm related setting called **DC terminal after PF alarm** (or similarly named) which, if set to **Auto**, let's the device automatically switch DC back on after PF has gone, so the control unit could poll the DC status to learn when the remote control can continue. Should this automatic feature not be in use, the control unit has to a) clear the PF alarm and b) wait for at least another 12 seconds before it can switch DC on again.

4. The ModBus protocol

4.1 General information about ModBus RTU

A telegram, as defined by the ModBus RTU protocol specification, consists of hexadecimal bytes of which the first byte isn't used by our devices as slave address in the original definition, because not needed for USB or Ethernet. However, the first byte must not be an arbitrary value, because it's used, amongst other purposes, to distinguish the message between ModBus and SCPI. Following are the permissible values in byte 0 for the series covered in this document:

Series	Allowed values for byte 0	Extra
All **10000** series	0x00, 0x01	0x01 is accepted, but only if the communication setting **ModBus specification compliance** is set to **Full**. By default,the setting is **Limited**. See the user manual of the device.
BT 20000	0x00, 0x01	-
BT 20000 Triple	0x00, 0x01, 0x02, 0x03	0x01-0x03 are to address the single channels. For more see below. 0x00 is used for broadcast queries to all channels.

A value between 0x02/0x04 and 0x39 in the first byte will cause a ModBus communication error, whereas from 0x3A (ASCII character: *) the telegram is considered as text message, means as an SCPI command.

Format and length of a ModBus telegram are defined. The telegram has to be transmitted according to the specifications of the particular interface that is used. Normally, the user only has to take care for a correct message, rather than correct transmission. But there are also interfaces, like for example RS232, which don't feature communication safety and don't guarantee flawless transmission. Other interfaces support flawless transmission by using a checksum and/or software handshaking.

4.2 General information about ModBus TCP

The message format according to the ModBus TCP specification is supported by the Anybus interface modules **ModBus TCP 1 Port** and **2 Port** (see section *2.1* for details), as well as by many device series with rigidly installed Ethernet port and from a specific firmware version. In both cases the default ModBus TCP port **502** has to be used.

Port 502 is reserved for ModBus TCP and thus blocked for "normal" Ethernet interface modules. The ModBus TCP interface modules, as well as the built-in LAN port of some series offer two sockets, the reserved port 502 and the adjustable port with default value 5025. It means that port 502 is automatically present and doesn't have to be set up.

By definition, a ModBus TCP message requires an additional header, compared to ModBus RTU. This makes it impossible to use SCPI via this port. The rest of the message is identical to ModBus RTU, except for the not required checksum. The below sections are related to the core part of ModBus messages, which is identical for both, RTU and TCP. Further ModBus TCP related information can be found in *"4.8. ModBus TCP in detail"*.

4.3 Format of set values and resolution

The value range of registers which can be set and/or read is stated in the series related ModBus register lists, if required. Most values are defined as 16 Bit, others as 32 Bit or even as float.

Set values for voltage, current, power or resistance are always directly settable 16 Bit values in form of a per cent value of the device's rated values of U, I, P and R and correspond at 100% to the hexadecimal value 0xCCCC (decimal: 52428). The total settable range is typically 0%...102% (0x0000...0xD0E5).

It means, you can set a per cent value between 0% and 100% by sending hexadecimal values of 0x0000-0xCCCC or for supervision thresholds of device alarms like OVP it will be 0x0000-0xE147 for 0% to 110% or with some series 0x0000-0xD2F1 for 0 to 103%.

This means 52429 possible values for 0-100%. This is internally halved (the MSB is reserved for sign), **so the effective resolution between 0 an 100% results in 26214 steps** or less. Regarding the topic "resolution" also see section *"3.8. Effective resolution when programming"*.

4.4 Translating set values and actual values

Real values have to be translated to per cent values before transmitting them to the device, as well as per cent values read from the device are usually translated into real values in order to process them further. Rule: 0xCCCC (hexadecimal) = 52428 (decimal) = 100% nominal value.

Translation is done by implementing these formulas into custom software:

Per cent value to real value	Real value to per cent value
Real value = $\frac{\text{Rated value * per cent value}}{52428}$	Per cent value = $\frac{\text{52428 * real value}}{\text{Rated value}}$
Example: The nominal voltage of your device is 80 V and actual voltage was read as 0x2454 (decimal: 9300). According to the formula above, the real actual value will be (80 * 9300) / 52428 = 14,19 V.	Example: the power set value shall be 3150 W, the power rating of your device is 3500 W. According to the formula above we get a power set value of (52428 * 3150) / 3500 = 47185 = 0xB851.

All set values are not only limited by the device’s rated values, but can also be limited by the adjustable “Limits”! Values exceeding the minimum or maximum of the limit range are rejected by the device.

When translating real values into per cent values (decimal or hexadecimal), it's often required to round up or down. We recommend to round naturally. Note that natural rounding can result in a translation value which is by 1 higher than expected.

4.5 Communication with the device

4.5.1 Ethernet

Continue to read in section *4.6*.

4.5.2 Profinet / Profibus

ModBus isn't supported directly via these field buses, but the addressing scheme of objects using slot and index, as well as the value format are connected to ModBus and the ModBus register lists. It can be considered as "indirect ModBus".

The Profinet/IO module (1 or 2 ports) or the Profibus module for devices with Anybus interface slot can be used to control and monitor a device using a network system, usually combined with an integrated PLC and proper software. For Profinet, there is no need to configure anything communication related on the device. Device name and IP related parameters are usually assigned from the master's location. The implementation of the device into a Profinet or a Profibus is described in section *"6. Profibus & Profinet"*. Continue to read in section *4.6*.

4.5.3 CANopen

ModBus isn't supported directly via this field bus, but the addressing scheme of objects is shifted into the manufacturer's range from index 0x2001, as well as the value format is connected to ModBus and the ModBus register lists. It can be considered as "indirect ModBus". Continue to read in section *4.6*. Also refer to section *"7. CANopen"*

4.5.4 CAN

ModBus isn't supported directly via this field bus, but the addressing scheme of objects is shifted into the manufacturer's range from index 0x2001, as well as the value format is connected to ModBus and the ModBus register lists. It can be considered as "indirect ModBus". Continue to read in section *4.6*. Also refer to section *"8. CAN / CAN FD"*.

4.5.5 ModBus TCP

The protocol used here is standard ModBus in a ModBus TCP frame. TCP/IP transmission itself isn't explained in this document. Continue to read in section *4.6*, plus also *"4.8. ModBus TCP in detail"*.

4.5.6 EtherCAT

ModBus isn't supported directly via this field bus, but the addressing scheme of objects is shifted into the manufacturer's range from index 0x2001 with 10000 series or into the user range from index 0x8000 with 20000 series, as well as the value format is connected to ModBus and the ModBus register lists. It can be considered as "indirect ModBus".

EtherCAT uses proprietary software (Beckhoff's TwinCAT or similar) and usually also SDO transfers for "CANopen over Ethernet" (CoE) protocol, which is a feature that required for our devices. Continue to read in section *4.6*. Also refer to sections *"7. CANopen"* and *"9. EtherCAT"*.

4.5.7 RS232

The RS232 module supports a variable baud rate, but the remaining serial settings are fixed: Data bits: 8, Stop bits: 1, Parity: none. Continue to read in section *4.6*.

4.5.8 USB (COM)

After connecting the device via USB cable and successful USB driver installation, the device is ready for access. The COM port, which is assigned to the new USB device (see Windows device manager) doesn't need configuration. It's based upon a so-called CDC driver (Communication Device Class), which is available for Windows 7 (also Embedded), 10 and most likely 11, as well as for other operating systems, too. This driver generates the COM port for simplicity and can run data transmissions as fast as the USB 2.0 port on the device can handle it. The typical serial settings are here effective and ignored by the driver which also means they're not required to be set specifically or, should the driver explicitly require it, would accept any typical values.

4.5.8.1 USB driver installation

The USB driver for the rear or front side type B port is included with the device on USB stick as installer. It installs a signed driver for virtual COM ports on 32 bit or 64 bit Windows operating systems since Windows 7. Alternatively it's available as download from our website. For Embedded OS versions of Windows the USB stick contains INF files.

4.5.8.2 First steps

In order to communicate with the device via USB, it actually only requires a software on the PC side which is able to open a COM port and send messages in either binary format (ModBus protocol) or as ASCII strings (SCPI). For the latter one, simple terminal softwares suffice, at least those which can send the complete command at once. For binary telegrams in hexadecimal format other tools are required, such as Docklight (www.docklight.de). We can provide ready-to-use example project files for Docklight upon request. Those can help for a start and to see how the communication works or if it works at all. The project files contain a few basic messages in form of macros which can be sent by the click of a button.

Concluding, to finally establish communication and access the device via USB, you just need to...

1. connect the device via USB.
2. install the USB driver (see *4.5.8.1*).
3. run a terminal or similar program.

4.6 About the register lists

Together with this programming guide, there are so-called register lists (usually one for each device series) included as PDF files. These lists give an overview about the remote programming features that are available for a certain device series when using binary communication protocols like ModBus, for which the lists are primarily made. Apart from that, they are also a substantial reference when controlling a device via a **field bus** (CAN, CANopen, Profibus, Profinet, EtherCAT) or accessing it in programming environments like **LabView** or MatLab, for example when trying to interpret values or to understand the function of a certain command.

The lists explain in compact format how the data in a binary message has to be interpreted or how a register (with CANopen or EtherCAT it's called "index") is specified. This will help the user to implement the device communication into custom software applications. Users who decide to work with SCPI command language usually don't need those lists. Later in this document, the SCPI commands are referenced in a separate chapter.

4.6.1 Columns "ModBus address"

This number, given in decimal and hexadecimal form, is the so-called ModBus register address or register number. It's used 1:1 and in hexadecimal form in ModBus messages.

4.6.2 Columns "Function"

The heads of the 5 columns contain the names and codes of the supported ModBus functions. An "x" in these columns marks the assignment of a register to any of the functions. For example, the so-called coil registers are usually writable and readable, so they're assigned to functions "Read Coils (0x01)" and "Write Single Coil (0x05)".

4.6.3 Column "Data type"

Data type	**Length**	
char	1 Byte	Single byte, used for strings
uint(8)	1 Byte	Single unsigned byte
uint(16)	2 Bytes	Double byte, also called word or unsigned 16bit integer
uint(32)	4 Bytes	Double word, also called long or unsigned 32bit integer
float	4 Bytes	Floating point value according to IEEE745 standard

4.6.4 Column "Access"

This column defines for every register whether the access is read only, write only or read/write.

R = Register is read only

W = Register is write only or wouldn't return a reasonable value when read from

RW = Register can be read or written

❗	*It applies generally: Writing to a register which allows write access (W, RW) is only possible during remote control!*

4.6.5 Column "Number of registers"

With ModBus, a register always has a length of 2 bytes or a multiple of 2 bytes. This column tells how many 2-byte values are used by the register. The value is always the half of the value in column “Data length in bytes”.

4.6.6 Column "Data"

This column tells additional information about the data which can be written to or read from the register. Two, four or more bytes can be interpreted in different ways, depending on data type.

4.6.7 Columns "Profibus slot/Profinet subslot" & "Profinet index"

These columns are only available in register lists for those series which support the optionally available Anybus interface modules, here in particular the Profibus and Profinet interfaces, or offer built-in Profinet/Profibus ports.

The columns are used by Profibus/Profinet users to link the registers in the list via the two values „slot” or „subslot” and also "index" to data blocks (SFBs) in the PLC software. For more details refer to *"6. Profibus & Profinet"*.

4.6.8 Column "EtherCAT SDO/PDO?"

This column is only available in register lists for those series which support the optionally available Anybus interface modules, here in particular the EtherCAT interface, or offer built-in EtherCAT ports.

The column marks which of the ModBus registers can be accessed via EtherCAT bySDO transfer using CANopen over Ethernet (CoE) protocol. Some of the marked registers are used in the PDOs while all marked registers are accessible as SDOs. Devices supporting EtherCAT contain a downloadable data object list. Which of the registers are linked to the PDO and other things are described in section *"9. EtherCAT"*.

4.7 ModBus RTU in detail

As described in section *"3.4. Overview of the communication protocols"*, this protocol can be used with via the built-in USB or Ethernet port, plus with some of the optionally available AnyBus modules for 10000 series. The addressed object when using the ModBus protocol is called **register**. This document uses the terms **address**, **register** or **register address**.

When transferring ModBus RTU messages via any Ethernet interface it's called "ModBus RTU over Ethernet", which isn't the same as "ModBus TCP". ModBus TCP frames are not supported via the built-in LAN port of the 10000 series, but via the optionally available ModBus-TCP interface modules. For more details refer to section 3.4.

4.7.1 Message types

Basically, the message system distinguishes between **query messages**, **control messages** and **response messages**. Query messages will cause the device to send a response message, while control messages only cause it to reply with a 1:1 echo, in order to confirm reception.

4.7.2 Functions

The second byte of a message contains a ModBus **function** code (**FC**, marked blue below), which determines whether the message is a READ or WRITE message. It also determines, whether one or multiple registers are accessed. The protocol, as described below, supports with date 02-27-2024 following ModBus functions :

Function code		**Function name**		**Description**	**Example of use**
Hex	**Dec**	**Long**	**Short**		
0x01	1	READ Coils	RC	Only allows to read 1 coil, because in our device series the coils are not organized incrementally. Earlier firmwares would always return 16 bits, ergo multiple coils, which was different from the ModBus specification, while only representing a boolean TRUE with 0xFF00 or FALSE with 0x0000. Also mind the note box in section *4*.	Query the input / output condition
0x03	3	READ Holding Registers	RHR	Used to read n subsequent registers. Results in n*2 bytes of data in the response message. Reading beyond a group of register is only possible is the next group is defined to have the same data type.	Read the model name string (1-40 bytes)
0x05	5	WRITE Single Coil	WSC	Used to write the coil (TRUE/FALSE) of a boolean register	Switch device to remote control.
0x06	6	WRITE Single Register	WSR	Used to write one register.	Set values (U, I, P etc.)
0x10	16	WRITE Multiple Registers	WMR	Used to write n subsequent registers. Can't be used to write beyond the limits of a register block, for example when trying to write multiple set values (U, I, P) at once.	Write multiple values at once within a register block or write the so-called user text

The register list defines which of the above functions may be used with every register.

The bytes in a ModBus message are read from left to right (big endian format), except for the 16 bit ModBus RTU checksum where low byte and high byte are switched.

4.7.3 Channel addressing with multi-channel devices

Multi-channel devices, such as those from series BT 20000 Triple, require a different approach in order to address the single channels and for that use the first byte of a ModBus RTU message. There is a distinction between single channel addressing (byte 0: 0x01 - 0x03 for channel 1-3) and simultaneous addressing of all channels (byte 0: 0x00, broadcast). As defined by the ModBus specification, broadcast isn't responded, so it can't be used for queries.

Custom made software, created for series that came onto the market before the 20000s in 2023, also including the 10000 series, cannot be used for 20000 series device "as is" and must be adapted. At least when software used to put a 0x00 into the first byte of a message.

The telegram description below all use address 0x01. This is valid for single channel devices, as well as multi-channel models where, in this case, channel 1 would be addressed.

4.7.4 Control messages (write)

The device checks the message only regarding the max. length of the register. After the data part, the checksum is expected. So in case the data part would only contain the minimum two bytes and thus the message would fulfill the protocol requirements for the selected function code, the checksum would be expected at the position of the 7th byte. If there were further data bytes at that position or zeros and the checksum would be at a different position in the message, the device would return an error. Hence the device will return an error, no matter if the telegram is too short or too long, because the checksum is wrong. For message examples see *"4.7.8. Examples of ModBus RTU messages"*.

WRITE Single Register

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Channel	**FC**	**Start reg.**	**Data word**	**CRC**
0x01	0x06	0...65535	Value to write	Checksum ModBus-CRC16 (1

WRITE Multiple Registers

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Byte 6	Bytes 7-253	Last 2 Bytes
Channel	**FC**	**Start reg.**	**Number**	**Count**	**Data bytes**	**CRC**
0x01	0x10	0...65535	0...123	Number*2	Data	Checksum ModBus-CRC16 (1

WRITE Single Coil

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Channel	**FC**	**Register**	**Data word**	**CRC**
0x01	0x05	0...65535	0x0000 or 0xFF00	Checksum ModBus-CRC16 (1

4.7.5 Query message

When querying something from the device, the response is expected to be immediately and will be of varying length, but always of the same construction. For the query, the start register and the number of registers or coils to read are required. The base of the ModBus data format is a register, a 16 bit integer value, means a group of two bytes. Hence, when querying one register with function READ Holding Registers, the device will return two bytes and when querying two registers it returns 4 bytes etc. With READ Coils, the response will be one byte (=1 coil) or two bytes (=16 coils, former response in earlier firmwares). For message examples see *"4.7.8. Examples of ModBus RTU messages"*.

READ Holding Registers

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Last 2 Bytes
Channel	**FC**	**Start reg.**	**Number**	**CRC**
0x01	0x03	0...65535	Number of regs to read (1...125)	Checksum ModBus-CRC16 (1

READ Coils

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Last 2 Bytes
Channel	**FC**	**Start reg.**	**Number**	**CRC**
0x01	0x01	0...65535	Must always be 1	Checksum ModBus-CRC16 (1

(1 See *"4.7.7. The ModBus checksum"*

4.7.6 Response message (read)

A response from the device is usually expected after a query or if something has been set and the device confirms the execution.

Expected response for WRITE Single Register:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Last 2 Bytes
Channel	**FC**	**Start reg.**	**Data**	**CRC**
0x01 (- 0x03)	0x06	0...65535	Written value echoed	Checksum ModBus-CRC16 (1

Expected response for WRITE Single Coil:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Last 2 Bytes
Channel	**FC**	**Start reg.**	**Data**	**CRC**
0x01 (- 0x03)	0x05	0...65535	Written value echoed	Checksum ModBus-CRC16 (1

Expected response for WRITE Multiple Registers:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Channel	**FC**	**Start reg.**	**Data**	**CRC**
0x01 (- 0x03)	0x10	0...65535	Number of written registers	Checksum ModBus-CRC16 (1

Expected response for READ Holding Registers:

Byte 0	Byte 1	Byte 2	Bytes 3-252	Last 2 Bytes
Channel	**FC**	**Data length in bytes**	**Data**	**CRC**
0x01 (- 0x03)	0x03	2...250	Queried registers content	Checksum ModBus-CRC16 (1

Attention! Mind the note box in section "4. The ModBus protocol" and the information therein.

Expected response for READ Coils (old format, compliance setting "Limited", default):

Byte 0	Byte 1	Byte 2	Bytes 3+4	Last 2 Bytes
Channel	**FC**	**Data length in bytes**	**Data**	**CRC**
0x01 (- 0x03)	0x01	2	0xFF00 or 0x0000	Checksum ModBus-CRC16 (1

Expected response for READ Coils (new format, compliance setting "Full"):

Byte 0	Byte 1	Byte 2	Byte 3	Last 2 Bytes
Channel	**FC**	**Data length in bytes**	**Data**	**CRC**
0x01 (- 0x03)	0x01	1	0x00 or 0x01	Checksum ModBus-CRC16 (1

Unexpected response (communication error):

Byte 0	Byte 1	Byte 2	Last 2 Bytes
Channel	**FC**		**CRC**
0x01 (- 0x03)	0x80 + Function code	Error code	Checksum ModBus-CRC16 (1

A communication error can have several reasons, like a wrong checksum or when attempting to switch a device to remote control that has been set to “Local” or if it's already remotely controlled by another interface. See the communication error code list in "4.9. ModBus communication errors".

4.7.7 The ModBus checksum

The checksum at the end of ModBus RTU messages is a 16 bit checksum, but isn't calculated as the usual CRC16 checksum. Furthermore, **the byte order** of the checksum in the message is **reversed**. Information about ModBus CRC16 and source code for implementation and calculation are available on the Internet, for example here: http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf , section 2.5.1.2.

(1 See *"4.7.7. The ModBus checksum"*

4.7.8 Examples of ModBus RTU messages

The payload of these examples can also be used with ModBus TCP.

4.7.8.1 Writing a set value

Set values are adjustable limits for the physical values Current, Voltage, Power and Resistance (where featured). The can only be written to a device, if it has been switched to remote control before via a digital interface.

Example: You want to set the current to 50%. According to the register list of a PSB 10000, the „Source mode: Set current value" is at address 501 (0x1F5) and the assigned function would be WRITE Single Register (WSR). Expecting the device to already be in remote control mode, the message to build then has to be like this:

Message to send:

Ch.	FC	Start	Data	CRC
0x01	0x06	0x01F5	0x6666	0x338E

▶ Expected response:

Ch.	FC	Start	Data	CRC
0x01	0x06	0x01F5	0x6666	0x338E

In this case, the device is expected to return an echo of your message, indicating successful execution of the command. The display of the device should now show 50% of what's the maximum current of your device. For a power supply or electronic load with 510 A nominal current it should show 255.0 A or for a model with 170 A current rating it should show 85.00 A.

4.7.8.2 Query all actual values at once

The device holds three readable actual values of voltage, current and power. Electronic loads feature an additional actual resistance value in their displays, which can't be read via interface, but is calculated from the actual voltage and current. Hence the user can calculate the actual resistance himself.

Actual values can be queried separately or all at once. The advantage of a combined query is, that you gain a snapshot of the most recent actual values of the DC input or output. When querying separately, values may have changed already when sending the next query.

According to the register list, the actual values start from register 507. Three registers shall be read:

Message to send:

Ch.	FC	Start	Data	CRC
0x01	0x03	0x01FB	0x0003	0x75C6

▶

Possible response:

Ch.	FC	Len	Data	CRC
0x01	0x03	0x06	0x2620 0x0C9B 0x091B	0x9350

4.7.8.3 Read the nominal voltage of a device

The nominal voltage, like the other nominal values of current, power or resistance, is an important value to read from a device. They're all referenced for translating set values and actual values. It's recommended to read them from the device right after opening the digital communication line, unless the software shall not be universal.

According to the register list, the nominal voltage is a 4-byte float value in register 121.

Query message:

Ch.	FC	Start	No.	CRC
0x01	0x03	0x0079	0x0002	0x15D2

▶ Possible response:

Ch.	FC	Len	Data	CRC
0x01	0x03	0x04	0x42A00000	0xEE69

Also see *4.7.6*. The response contains a float value according to IEEE754 format, which translates to 80.0.

4.7.8.4 Read device status

All device report their device status in register 505.

Query message:

Ch.	FC	Start	No.	CRC
0x01	0x03	0x01F9	0x0002	0x15C6

▶ Possible response:

Ch.	FC	Len	Data	CRC
0x01	0x03	0x04	0x00000483	0xB952

Also see *4.7.6*. The response contains the value 0x483 which states that the device is in remote control via the USB port, that the DC input/output is switched on and that CC (constant current) mode is active.

4.7.8.5 Switch to remote control or back to manual control

Before you can control a device from remote, it's required to switch it to remote control. This is done by sending a certain command.

❗	*The device will never switch to remote control automatically and can't be remote controlled with being in this condition. Reading from all readable registers is always possible.*

❗	*The device won't exit remote control automatically or at least only for one specific reason: the interface monitoring feature, which has to be activated willingly and explicitly by the user. One part of the defined reaction to the elapsed timeout is to leave remote control. Refer to the user manual of the device for more information. Usually, remote control is left anytime by sending a certain command or simply switching the device off.*

Switching to remote control may be inhibited by several circumstances and is usually indicated by an error message:

- Condition **Local** is active (check the display on the front of your device or read the device status), which will prevent any remote control
- The device is already remotely controlled by another interface
- The device is in setup mode, means the user has accessed the setup menu and not left it yet

► How to switch a device to remote control:

1. When using the ModBus RTU protocol, to create and send a message according to the description above, for example 01 05 01 92 FF 00 2C 2B.
2. Once the switchover to remote control has been successful, the device should indicate the new condition in the display or with an LED, as well as it echoes the message as confirmation.

In case switching to remote control would be denied by the device, because the setting **Allow remote control** in the HMI of the device is set to **No**, then the device will return an error message like 01 85 17 02 9E. According to the ModBus specification, this is error 0x85 with error code 0x17.

Leaving remote control can be done in two ways: using the dedicated command or by switching the device to **Local** condition. We will consider the first option, because this is about programming.

► How to exit remote control:

1. When using the ModBus RTU protocol, you need to build and send a message according to the description above, for example 01 05 01 92 00 00 6D DB.
2. Once the return to manual control has been successful, the device should delete the status "remote" from the display, as well as it echoes the message as confirmation.

4.8 ModBus TCP in detail

This section is only about the differences of TCP message format compared to RTU message format. The core of a ModBus TCP message is still ModBus RTU. Section *"4.7. ModBus RTU in detail"* holds more information. Overview:

- A ModBus TCP message requires an additional 6, actually 7 bytes long MBAP header
- The checksum is omitted
- Transmission only via reserved port 502; any other port won't accept ModBus TCP frames
- Will only be used with Ethernet and TCP, contrary to ModBus RTU which can used on many different lines

As a result, Modus TCP messages are always 4 bytes longer than **ModBus RTU** messages. The MBAP header is specified as follows:

Bytes	Meaning	Explanation
0 + 1	Transaction identifier	This identifies the message. It's copied by the device in the response and is used to identify a certain message in a pool of incoming transmissions if multiple device are communicating with the PC and the response isn't immediately. The identifier is an arbitrary value between 0 and 65535.
2 + 3	Protocol identifier	Here always 0 = ModBus protocol
4 + 5	Length	Number of remaining bytes in the message, i.e. the length of the ModBus RTU core message minus 2.
6	Channel Unit identifier (UID)	This byte is used by the ModBus TCP slave as channel selector. Single channel devices may receive 0x00 or 0x01 here, while multi-channel device support to receive 0x00-0x03, whereas 0x00 would be interpreted as broadcast to all channels at once.

4.8.1 Example for a ModBus TCP message

The example for READ Holding Registers from section *"4.7.8.3. Read the nominal voltage of a device"*, extended by the MBAP header and an arbitrary transaction identifier of 0x4711 and addressing channel 2 of a multi-channel device. This message would be expected to return an error frame with a single channel device:

Query message:

MBAP header	Ch.	FC	Start	Length
0x4711 0x0000 0x0006	0x02	0x03	0x0079	0x0002

Possible response:

MBAP header	Ch.	FC	Length	Data
0x4711 0x0000 0x0007	0x02	0x03	0x04	0x43FA0000

With this, the control unit would query the device's nominal voltage. The example response contains a floating point value in message part **Data**, which translates to 500 and since this has been queried as the rated voltage, it means 500 V.

4.9 ModBus communication errors

Communication errors are only related to digital communication with the device. Other alarms or errors of any kind which can be generated and indicated by the device must not be mixed up with these.

The device will return unexpected error messages in case the previously sent message is in wrong format or if the function can't be executed for some reason. For example, when trying to write a set value with WRITE SINGLE REGISTER while the device isn't in remote control. Then the message isn't accepted and the device will return an error message instead of a confirmation message. The message format can be wrong if the checksum is bad or if you try to read a bit with function READ Holding Registers instead of READ COILS.

In case of an error, the response message contains the original function code added to 0x80, in order to identify the response as error message. Overview of function codes in error messages:

Error FC	Belongs to
0x81	READ COILS
0x83	READ HOLDING REGISTERS
0x85	WRITE SINGLE COIL
0x86	WRITE SINGLE REGISTER
0x90	WRITE MULTIPLE REGISTERS

Overview of the communication error codes which can be returned by the device:

Code		Error	Explanation
0x01	1	Wrong function code	The function code in the 2nd byte of the ModBus message isn't supported. See *"4.7.2. Functions"* for supported codes. The error also occurs using the wrong function code on a register
0x02	2	Invalid address	Various causes: • The register address isn't defined for your device. Every device series can have a different number of registers. Refer to the separate ModBus register list of the series your device belongs to. • When using ModBus RTU and slave address 0x01 with an older KE firmware where it's only supported to use address 0x00 or when the switch "ModBus specification compliance" is set to "Limited".
0x03	3	Wrong data or data length	The length of data in the message is wrong or the data itself. For example, a set value always requires two bytes of data. If the data part of the message would be one byte only or three bytes, then the data length would be wrong. Otherwise, when sending a set value of, for example, 0xE000 to a register for which the maximum value is defined as 0xCCCC, this would be wrong data.
0x04	4	Execution	Command could not be executed, depends on the situation
0x05	5	CRC	The CRC16 checksum at the end of the ModBus RTU message is wrong or missing (perhaps due to split TCP packets) or has been transmitted in wrong byte order (high byte first instead of low byte)
0x07	7	Access denied	Access to a certain register isn't allowed or read only while trying to write, or vice versa, depending on the current device situation. Example: you cannot write to register 403 while the device isn't in remote control mode or while it's under remote control from a different interface.
0x17	23	Device in local	Indicates, that write access to the device is blocked by the "local" condition, so only read access is possible. "Local" means that remote control isn't allowed.

An example: there has been an attempt to switch the device to remote control, but instead of an echo of the message it returns something like this: 01 85 07 03 52. This is an error message. The position of the function code contains the value 0x85. According to the first table above, this is related to the function WRITE SINGLE COIL. The error code in the message is 0x07 which means, according to the second table above, the device has denied the access. This can have different reasons, but the most likely one is that the device is already in remote control via a different interface.

4.10 Explanation about specific registers

Reminder: for the abbreviations of the devices series see *"1.1.2. Validity"*.

Many of the commands or register related options are self-explaining, but not all of them. Some of the not self-explaining ones will be handled below. Not all series feature the same number of commands. Every example below will indicate to which series it's compatible by showing a small table:

10K	BT
✓	–

whereas

✓	means, the command or the command group (entirely or partly) is supported by the device series.
–	means, the command or the command group isn't supported by the device series.

4.10.1 Register 171

This allows to write and read an arbitrary string of up 40 characters, which can be used to uniquely identify a device amongst multiple units of the same model. It's permanently stored after being written.

4.10.2 Register 411

Described for SCPI in *"5.4.14. Commands for alarm management"*.

When using ModBus, this register is intended to reset alarm bits as represented in the device status (register 505, see below). Until these are not reset, which is considered as an acknowledgment, the bits from previously occurred alarms remain set, even if the alarms are gone already. Alarms which are still present while register 411 is used to reset the alarm bits will of course be excluded from resetting. After resetting the alarm bits, device alarms can later only be read in form of an alarm counter (registers 520 - 524).

4.10.3 Registers 500-503 (set values)

These are the most important registers to work with, because they define the DC output/input values of voltage, current, power and resistance (where featured). With ModBus, any set value is transmitted as per cent value of the nominal device values (0...100%), whereas for SCPI real values are used.

Generally, before you can use R mode with devices where internal resistance is featured, it has to be activated (register 409), otherwise the set value is ignored.

For power supply series which feature the so-called "simple" PV function, the set value for current (register 501) isn't interpreted as current, but as factor on the current, representing the irradiation. It can be adjusted in 1% steps, meaning 100 steps between "total darkness" and "bright light"

4.10.4 Registers 498, 499 and 504 (additional set values)

Three additional set value registers for the so-called sink mode operation of bidirectional device series, such as PSB 10000, PSBE 10000 and BT 20000. There are 498 (sink power), 499 (sink current) and 504 (sink resistance).

4.10.5 Register 505 (1. Device status register)

Another important register, as it represents the most relevant device condition bits in one 32 bit value (ModBus). Some bits are grouped and have to be interpreted as such. According to the register lists, bits 0-4 of register 505 are a group that represents the so-called control location (see *"3.2. Control locations"*). By reading this register you can furthermore detect if the device is still in remote control.

With SCPI, some but not all of these 32 bits of this register are represented in the status registers "Questionable" and "Operation". See *"5.4.2. Status registers"*.

4.10.5.1 When running master-auxiliary

During master-auxiliary (formerly: master-slave) operation, the status register uses bit 29 ("MAS") as an additional alarm to indicate the so-called master-auxiliary safety mode, which is activated every time the master detects any problem in the communication with the auxiliary unit(s). This can occur due to a connection failure or heavy electrical interferences. The master unit will then set this bit and switch off all DC terminals of the aux units that are still accessible. Offline aux units will put themselves into a similar state and switch DC off. After removal of the problem cause, the MS system can be re-initialized, which will also clear the MAS bit.

4.10.5.2 Bit 31

Only featured with bidirectional device series (PSB, PSBE, BT), this bit indicates the actual state regarding source or sink mode. It's used to determine whether a recently read actual value of current or power belongs to source mode operation (current flows out of the device) or sink mode operation (current flows into the device).

4.10.6 Register 511 (2. Device status register)

Further alarm and device condition bits, partially reserved for future software features.

4.10.7 Registers 650 - 662 (master-auxiliary configuration)

10K	BT
✓	✓

This block of registers is used to configure the master-auxiliary operation mode (short: MA) the same way as it can be done in the setup MENU of your device. Refer to the device's operating guide about how the MA works and what do to in preparation of its remote control. For remote control of a MA system, it's expected to be fully wired. Before MA operation, slave units can be configured remotely, but during MA operation they can only be monitored, if required. It's, however, recommend to only control the master unit. Configuration and activation of MA operation can also be done manually and remote control can be taken over later after the master has initialized the system.

With the MA system not being set up yet, these registers have to be used in a certain order on any unit:

1. Switch to remote control with register 402.
2. Activate MA operation mode with register 653.
3. Select with register 650 whether the unit you are configuring will be **Master** or **Auxiliary**.

Further steps, only to be performed on the master unit:

4. Initialize the MA system with register 654.
5. Optional: check with register 655, whether the initialization has been successful.
6. Optional: Query the number of initialized auxiliary units with register 662 --> in case the returned number doesn't match the number of auxiliary units you want to use in the MA system, check the settings of all units and the cabling and repeat the initialization.
7. Optional: read the nominal values (registers 656-660) of the previously initialized MA system to be used as value translation reference while running the MA.
8. Optional: configure alarm thresholds, event thresholds and set value limits.

During MA operation, the remotely controlled master unit can be accessed like a single unit, with a few exceptions (see device manual). Set values and actual values are always per cent values in relation to the ratings. Access to those set values registers is described in the other sections.

4.10.8 Registers 850 - 6695 (Function generator)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

The integrated function generator is a complex feature. It's configured and loaded with a lot of registers. Before you can run a function, setup is required every time and in a certain order. Those standard functions like rectangle or sine wave, as directly available on the HMI, are only indirectly accessible in remote control, via the arbitrary generator

First of all, you need to decide which one of the two generators you want to use, **arbitrary** or **XY**. All further steps depend on this selection. Other functions, such as battery test or MPP tracking, belong to the function generator, but are internally realized by code.

All function generator settings and loaded data (sequence points, XY table) are not stored inside the device and have to be loaded at least once after powering it. These data and settings are separate from what you can manually set up on the HMI.

4.10.8.1 Procedure for the arbitrary generator

This generator is used to create wave functions like sine, square, triangle or trapezoidal.

Step 1:

Select, whether to apply the function to the voltage U (register 851) or the current I (register 852). Before you haven't made this selection, the device can't accept sequence point data, because the sequence data is run through a plausibility check against the device's adjustment limits, which might lead to an error message and rejection of the sent data.

Step 2:

Define start sequence point (register 859), end sequence point (register 860) and number of cycles of that block to repeat (register 861).

Step 3:

Load data for x out of 99 sequence points (registers 900-2468, 8 values per sequence point).

In case of an unexpected error when writing sequence point data, though the parameters are seemingly OK, it most likely has at least one of these reasons:

a) CANopen, CAN or EtherCAT are used as communication line where the 8 parameters of a sequence point are not transferred as one block. Here the order of written parameters might require to write the time value (8th parameter) first, because in the internal memory of the KE component this value isn't initialized to its minimum value and thus the plausibility check would run against a time of 0.

b) A change in the AC or DC values between start and end is desired which, together with the time value would result in a certain slope that doesn't meet the minimum. Since a minimum slope isn't a topic anymore in the 10000 series since a certain firmware release, it's not dealt with in this document, but there is a section in the user manual of the device for more information. In case of a frequency sweep see "3.9. Minimum frequency slope (arbitrary function generator)".

Step 3.1:

Not always required: send submit command (register 862). This would always be required when using an interface which can't send the data of a sequence in one message, which would be CAN, CANopen and EtherCAT.

Step 4:

Set the global voltage limit (register 500), should the function be applied to the current. Otherwise, set the global current limit (register 501), in case the function is applied to voltage. Set the global power limit with register 502. For a bidirectional device additionally set 498 and 499.

Step 5:

Control the function generator with start/stop (register 850), which would automatically turn the DC input/output on if still off. Alternatively and when it's wanted to settle the static values before the actual function run, the DC input/output could be switched on (register 405) separately and before starting the function. After stopping the function run, the FG can also be reconfigured like when just selecting a different block of sequence points to run by changing start and end sequence point. This would, however, require to send the submit command again for specific interfaces. See step 3.1.

Step 6:

When finished, leave the function generator by deselecting either U mode (register 851) or I mode (register 852).

4.10.8.2 Programming example for the arbitrary generator

Before you can configure the arbitrary generator for a ramp or another wave type it's necessary to think about the best way to achieve the ramp generation. It's important to keep in mind that the arbitrary generator stops at the end of the function run, unless you set the repetition to infinite. After this kind of stop, the DC input/output remains switched on. In case of a ramp, this is wanted, because the end value shall usually remain set for time x. However, the device will go to static mode again, setting the static set values of U, I and P. The static values also apply for the period before the function run and for situations when the DC output/input is already switched on. In the particular case of a ramp it would require to have a different static value of, for instance, voltage before and after the function, which isn't possible.

The stop action and the static values are thus a little problematic for the ramp function. Why? Supposing you wanted to have a power supply generate a ramp starting from 0 V. The static value for U (voltage) would then be set to 0. But after the function stop, the device would also set 0 V and the voltage would drop from whatever ramp end value has been reached during the function run. Conclusion: the static value of voltage after the ramp end has to be part of the function.

In order to achieve this, the function has to consist of two parts: one for the rising or falling ramp and the other for the static value. This can be done using two sequences of the arbitrary generator.

Assumption: you have a power supply and the ramp shall start from 0 V and rise to 50 V within 6 seconds. The end voltage shall remain constant for 3 minutes (the time can be varied at will). Sequences 1 and 2 will be used. Remote control is already active, we only need to configure the sequences. Since the ramp will make the voltage rise linearly, only the DC part of a sequence suffices while the parameters for the AC part (indexes 0 - 4) should be set to zero in order to avoid remainders from perhaps previous configurations, which could disturb the correct wave generation.

The first step is to **activate function generator mode**, in this case we select arbitrary generator for U:

Ch.	FC	Start	Data	CRC
0x01	0x05	0x0353	0xFF00	0x7C6F

Next step is to create the **ModBus message to configure sequence 1**, **the rising ramp**. According to the register list start register 900 (WMR, function code 0x10) is assigned to sequence 1. Because the data part wouldn't fit the width of this document's page size, the 8 float values are below each other:

Ch.	FC	Start	Regs	Bytes	Data	CRC	Description
0x01	0x10	0x0384	0x10	0x20	0x00000000		Start value of AC part: 0 V
					0x00000000		End value of AC part: 0 V
					0x00000000		Start frequency of AC part: 0 Hz
					0x00000000		End frequency of AC part: 0 Hz
					0x00000000		Start angle of AC part: 0°
					0x00000000		Start value of DC part: 0V
					0x42480000		Start value of DC part: 50V
					0x4AB71B00	0x52B8	Rise time in µs: 6,000,000 (6 seconds)

After this, the **ModBus message to configure sequence 2, the static voltage** would be next. Start register here is 916:

Ch.	FC	Start	Regs	Bytes	Data	CRC	Description
0x01	0x10	0x0394	0x10	0x20	0x00000000		Start value of AC part: 0 V
					0x00000000		End value of AC part: 0 V
					0x00000000		Start frequency of AC part: 0 Hz
					0x00000000		End frequency of AC part: 0 Hz
					0x00000000		Start angle of AC part: 0°
					0x42480000		Start value of DC part: 50V
					0x42480000		Start value of DC part: 50V
					0x4D2BA950	0x627B	Hold time in µs: 180,000,000 (= 3 minutes)

And as last step, configuration of the rest, means the function run and globals:

Ch.	FC	Start	Data	CRC	Description
0x01	0x06	0x035B	0x0001	0x399D	Register 859, WSR, start sequence: 1
0x01	0x06	0x035C	0x0002	0xC85D	Register 860, WSR, end sequence: 2
0x01	0x06	0x035D	0x0001	0xD99C	Register 861, WSR, sequence cycles: 1
0x01	0x05	0x035E	0xFF00	0xEDAC	Register 862, WSC, submit the settings for the FG now
0x01	0x06	0x01F5	0xCCCC	0xCD51	Register 501, WSR, global current limit: 100%
0x01	0x06	0x01F6	0xCCCC	0x3D51	Register 502, WSR, global power limit: 100%

Setting the global values (current, power) to maximum or any other value that wouldn't interfere the ramp generation is necessary, especially when running multiple devices in master-slave where those set values also limit the DC output/input working range of the auxiliary unit(s).

Now the entire function setup is done and the function can be started. When doing so and if the DC output/input of your device would still be off when starting the function, it will automatically switch on. Alternatively, you could switch it on separately with the corresponding command and before actually running the function. In this example it's not necessary, because the voltage shall start to rise from 0 V. In other situations where the starting level isn't zero, it would be required to switch on the DC output/input first and wait for the voltage to settle.

For the number of sequence cycles 1 is sufficient, but it can be changed at will. Then the entire function would be repeated after 3 minutes and 6 seconds. The voltage, when using a power supply, wouldn't instantly drop to 0 V at the end of the first function run and before the second one starts. It depends on the load how long the voltage takes to sink and the ramp, when being graphically recorded on an oscilloscope, could look different than expected. This could be avoided by adding a third sequence which only uses some time for the voltage to go down.

Ch.	FC	Start	Data	CRC	Description
0x01	0x05	0x0352	0xFF00	0x2DAF	Register 850, WSC, Run function

4.10.8.3 Procedure for the XY generator

Step 1:

Select the XY function generator mode with register 856. All modes are based on an IU curve:

Mode	Support			Load XY data starting at register
	ELR1	PSB1	BT	
IU / IU (Source)	✓	✓	✓	2600
IU (Sink)	–	✓	✓	40960
IU (Source)+ IU (Sink)	–	✓	✓	2600 and 40960
Simple PV (curve A)	–	✓	✓	2600
Simple PV (curve B)	–	✓	✓	40960
FC	–	✓	✓	2600

- *An ELR, as an electronic load, run natively in sink operation only, but the distinction between source and sink operation mode has only been created when the bidirectional series came up and since that was after ELR, the ELR today runs the XY generator in a mode that with a PSB is called "IU (Source)".*
- *The combined mode of IU for source and sink can be used for two-quadrants operation.*
- *The PV curves A and B can be loaded in parallel, but not hot-switched during simulation run*

Step 2:

Load the XY table data in 256 blocks of 16 values into registers 2600 - 6680 and/or 40960-45040, depending on the selected mode (see table above). This corresponds to max. 4096 values per table or LUT (lookup table) for a measurement range of 0-125% U_{Nom} or I_{Nom}. It means, the device measures its current flow and the voltage on the DC terminal in a range of 125%. The range above 102% can still be used to have the curve react to measurings, but the loaded table values may still not exceed 102% of the rating. The device will check all values and would reject excess values with a communication error. Less values than the default 4096 can also be loaded, for instance 3277 values for 0-100% if the application only wants to cover that range. All values which are not written will result in 0 V or 0 A.

Step 3:

Set static values which are not affected by the table. These are very important for master-auxiliary systems where they define the working range of the auxiliary units which else might not supply or sink power. Basically, voltage (register 500) and power (register 502) are set which, in case of bidirectional devices being used, would suffice for source only operation. Should sink only operation be run with a bidirectional device or a combination of both, sink and source, then register 498 would either be set alternatively to 502 or additionally.

Step 4:

Run the function generator by switching the DC input/output of the device or MA system on (register 405). During the function run, which in case of the simple PV mode is also called "solar panel simulation", you may want to control irradiation by sending set values to register 501, which usually is for current. 100% irradiation corresponds to a factor of 1 and 0% to a factor of 0. This factor is multiplied to the simulated current I_{MPP} of the MPP which usually is situated somewhere on the PV curve as loaded in step 2.

Step 5:

Exit the function generator by unselecting IU mode via register 856.

4.10.9 Registers 9000 - 9009 (Adjustment limits)

10K	BT
✓	✓

For SCPI, this is explained in *"5.4.8. Commands for adjustment limits"*. ModBus users should also read that section for the general handling of these settings. Apart from that, setting these parameters is like setting a set value (U, I, P, R).

4.10.10 Registers 10007 - 10900

10K	BT
✓	✓

Those registers can be used to remotely configure the various built- in or optionally available digital interfaces for the above stated series. The registers are connected to the corresponding settings in the device's setup menu.

Contrary to manual control, the settings for the pluggable interface modules of series IF-AB, as available the 10000 series can even be configured while the interface module isn't yet installed.

4.10.11 Registers from 11000 (MPP tracking feature)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	–

The MPP tracking feature is only available with devices that feature the "function generator"(FG), though MPP isn't connected to that FG. With this feature the device emulates the characteristics of a solar inverter device when seeking to find the maximum power point (MPP) of a solar panel. More details about this feature and the available modes are in the user manual of those series supporting this feature.

Registers 11000 - 11016 are related to the configuration parameters as you would adjust them on the display of the load device. Registers 11100 - 11199 are related correspond to the "load voltage values from USB stick" function when using manual control, while parameters 11200 - 11499 correspond to the "save the results to USB stick" function when using manual control and after MPP4 mode has been finished gathering data.

4.10.11.1 Programming the MPP4 mode

The table shows the sequence of commands to send in order to load and run a user defined curve with 75 points.

Register	Name	Purpose
11000	Select MPP4	Switch the function generator to mode MPP4
11000 - 11174	Load curve data	Load 75 voltage values that represent a user defined curve. The next two steps define the actual range of points to run through, which ideally matches the number of loaded points. In case a point is processed which has not been loaded, the device will set 0 V.
11015	Set end point	Defines the end point of a range of points to run through. Can be an arbitrary value between 1 and 100. Since the start point can't be higher than the end point, the end point is set first.
11014	Set start point	Defines the start point of a range of points to run through. Can be an arbitrary value between 1 and end point, because it can't be higher than the end point.
11013	Tracking interval	Defines the time (in milliseconds) between two curve points.
11016	Repetitions	Defines the number of additional cycles. The result data, which can be read later, will always contain the data from the last cycle. If the curve shall be run only once, set this to 0.
11010	Start/stop tracking	After the start of the MPP tracking, the device will set the voltage of the first point in the defined range, measure current and power and store the values. Then it would continue to the next point etc. This mode stops automatically after a duration which results from the tracking interval, the number of points in the range and the repetitions. The test can be stopped anytime with this register.
11011	Read status	optional: read the status of the tracking run in order to determine when it's finished
11012	Read errors	optional: read possible errors during or after the test to determine if the test has run through successfully

4.10.12 Registers from 11500 (battery test)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	–

Devices with the capability of running as electronic loads, such as series PSB 1000 or ELR 10000, and which feature a function generator also offer a battery test function which is also completely configurable and controllable via remote control. This battery test, when running on an electronic load, can only discharge a connected battery while with a power supply, particularly a bidirectional one, it can also charge. See the comparison table below.

The registers are connected to the parameters as shown on the HMI when manually operating this test function. Hence it's recommended to read the section about the battery test in the corresponding device manual and probably to "play" around with the settings on the HMI to get a feeling for the flow before starting with remote programming. A short overview about the involved registers:

Register	Purpose	ELR?	PSB/BT?
11535	Activates the battery test and selects the mode	x	x
11500 - 11513	Configuration of the static current battery test mode "Discharge"	x	x
11514 - 11531	Configuration of the pulsed current battery test mode "Discharge"	x	x
11545 - 11558	Configuration of the static current battery test mode "Charge"	-	x
11559 - 11584	Configuration of the dynamic battery test mode "Charge/Discharge"	-	x
11532	Test run control	x	x
11536 - 11544	Evaluation (time, Ah, Wh)	x	x

4.10.13 Register from 12000 (extended photovoltaics simulation acc. DIN EN 50530)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	–	✓

Photovoltaics simulation is a source mode function based on the XY generator. All ModBus registers which represent parameters related to this simulation and which can be written to the device or read from are referenced in the EN 50530 norm paper. The paper is furthermore the reference for the user regarding setup and correct use of this simulation feature.

The procedure to set up and control the extended PV simulation using ModBus protocol isn't different to manual handling (see user manuals of the devices) on the device's HMI or when using SCPI commands. See the examples in section *"5.5.3. Programming examples for PV simulation (DIN EN 50530)"*. These step by step examples have an extra column in the table that holds the related ModBus register number as cross-reference. One of these examples (nr. 2) converted to ModBus RTU format (percent set values translated for a device with 80 V and 170 A rating):

Configuration (before the start)

Nr.	Command	Description
1	01 05 01 92 FF 00 2C 2B	Activate remote control
2	01 06 2E E1 00 03 90 D5	Activate PV simulation mode **DAYET**
3	01 06 2E F0 00 00 80 D1	Select technology: Manual (all required parameters must be defined, here as with commands 4-10)
4	01 10 2F 02 00 02 04 3F 4C CC CD F3 11	Fill factor voltage (FF_U): 0,8
5	01 10 2F 04 00 02 04 3F 47 AE 14 EA 03	Fill factor current (FF_I): 0,78
6	01 10 2F 06 00 02 04 39 9D 49 52 80 AB	Temperature coefficient α for I_{SC}: 0,0003 /°C
7	01 10 2F 08 00 02 04 BB 44 9B A6 A5 83	Temperature coefficient β for U_{OC}: -0,003 /°C
8	01 10 2F 0A 00 02 04 3D 94 7A E1 04 89	Scaling factor C_U for U_{OC}: 0,0725
9	01 10 2F 0C 00 02 04 39 66 AF CD 7B 2D	Scaling factor C_R for U_{OC}: 0,00022 m²/W
10	01 10 2F 0E 00 02 04 3B 4E 70 3B A3 32	Scaling factor C_G for U_{OC}: 0,00315 W/m²
11	01 05 2E F1 FF 00 D4 E1	Select input mode: **ULIK**
12	01 06 2F 10 61 47 E9 79	Set open circuit voltage: 38 V (=0x6147)
13	01 06 2F 11 08 6F 96 F7	Set short-circuit current: 7 A (=0x086F)
14	01 05 2E F2 FF 00 24 E1	Activate data recording
15	01 05 2E E5 00 00 D5 15	Deactivate interpolation of day trend data
16	01 06 01 F4 61 47 A0 66	Set global voltage limit: ≥Uoc (=0x6147)
17	01 06 01 F6 CC CC 3D 51	Set global power limit: 100% (=0xCCCC)

Write day trend data (before the start)

Nr.	Command	Description
18	01 05 2E E6 FF 00 64 E5	Select access mode: write
19	01 05 2E E7 FF 00 35 25	Delete former data (should be executed every time before loading new data)
20	01 10 2E EA 00 06 0C 00 00 00 01 44 44 66 66 00 00 03 E8 B5 76	Write 1st day trend data set into index 1: Irradiation: 500 W/m² (=0x4444) Temperature: 20°C (=0x6666) Dwell time: 1000 ms (=0x000003E8)
	! *The dwell time is defined to have a minimum of 500 ms. However, for the <u>very first</u> day trend data set it's expected to set 1000 ms or higher, because else the function run might fail.*	
21	01 10 2E EA 00 06 0C 00 00 00 02 6D 3A 74 0D 00 00 05 DC D9 3F	Write 2nd day trend data set into index 2: Irradiation: 800 W/m² (0x6D3A) Temperature: 28°C (=0x740D) Dwell time: 1500 ms (=0x000005DC)
...		Write further data sets, a total of 500

Nr.	Command	Description
519	01 10 2E EA 00 06 0C 00 00 01 F4 A3 D6 7F FF 00 00 4E 20 09 53	Write 500th day trend data set into index 500: Irradiation: 1200 W/m² (=0xA3D6) Temperature: 35°C (=0x7FFF) Dwell time: 20000 ms (=0x000034AF)

Control

Nr.	Command	Description
520	01 05 2E E0 FF 00 84 E4	Start simulation (register 12000) -> the simulation will stop automatically after the time has elapsed that results from the total of dwell times in all written data sets

During the simulation, the index counter in register 12010 is updated with every next day trend point on the curve. It can be read and used to determine at which point the curve has been stopped due to an unexpected error, such as a device alarm.

Analysis after simulation end

Nr.	Command	Description
521	01 03 2E F4 00 02 8C D1	Read number (n) of recorded data sets. This number isn't related to the number of day trend data sets in use. This feature records a new data set every 100 ms. Depending on the total simulation time, the record buffer could fill (max. 16 h record time) and overwrite existing data. It may become necessary to calculate the total simulation time from the day trend data sets and start reading the recorded data during simulation, then clearing the buffer and later read the rest of data.
522	01 10 2E F6 00 02 04 00 00 00 01 68 A0	Select first data set (index 1) for reading
523	01 03 2E F8 00 08 CC D5	Read data from data set (index) 1
...		Read further n-1 data sets:

5. SCPI protocol

SCPI is an international standard for a clear text based command language. Details about the standard itself can be found on the Internet.

5.1 Format of set values and actual values

In the SCPI command language **real values** are used, with or without unit. It means, if you wanted to set a current of 177.5 A you would use the simple command **CURR 177.5** or, with unit, **CURR 177.5A**. Below you will find more detailed information about the available commands and their syntax.

5.2 Syntax

Specification according to „1999 SCPI Command reference”. Following syntax formats can occur in commands and/or responses:

Values	This numeric value corresponds to the value in the display of the device and depends on the nominal values of the device. Rules: - The value must be sent after the command and separated by a space - Instead of a numeric value you can also use:	
	MIN	corresponds to the minimum value of the parameter
	MAX	corresponds to the maximum value of the parameter
<NR1>	Numeric values without decimal place(s) and without physical unit	
<NR2>	Numeric values with decimal place (floating point), includes NR1, with or without physical unit	
<NR3>	Like <NR2>, but with multiplier (supported are: k/K for kilo)	
<NRf>	<NR1> or <NR2> or <NR3>, negative values supported	
Unit	V (Volt), A (Ampere), W (Watt), OHM, s (Seconds)	
<CHAR>	0..255: Decimal value	
<+INT>	0..32768: Positive integer value (output from device)	
<B0>	1 or ON: Function is/will be activated	
	0 or OFF: Function is/will be deactivated	
<B1>	NONE: manual operation active, switching to remote control possible	
	LOCal: local (manual) operation only, reading of data possible	
	REMote: device is in remote control	
<ERR>	Error with number and description	
<SRD>	String data, various formats: - IP address as number string with dots as separator, for example „192.168.0.2“ - Key words, for example AUTO or OFF	
<Time1>	<NR2>s (floating point time format in seconds)	
<Time2>	<SRD> as HH:MM:SS or HH:MM:SS.MS (hours/minutes/seconds/millseconds)	
<Time3>	<NR1> in seconds or milliseconds	
;	The semicolon is used separate multiple commands within one message	
:	The colon separates the SCPI keywords (main system, subsystems)	
[ ]	Lowercase letters and the content of square brackets are optional	
?	The question mark identifies a message as query. A query can be combined with a control message (command concatenation). Note, that it's required to wait for the response of the query before the next control message can be sent.	
->	Response from device	

5.2.1 Upper and lower case

SCPI uses upper case commands by default, though the device also accept lower case form.

5.2.2 Long form and short form

SCPI commands have a long form and a short form. The short form (e.g. SOUR) and the long form (e.g. SOURCE) can be used arbitrarily. To distinguish both forms, the commands as described in the following sections are written partly in upper case (indicates short form), partly in lower case letters (indicates the additional part of the long form).

5.2.3 Channel addressing

Multi-channel devices require a specific channel addressing for channels other than 1. This is done by adding a syntax to the original command which then would select a single channel or a group. It means, it becomes possible to switch the DC terminals of all channels of a BT 20000 Triple on or off at the same time, with one command. Or the status of channel 1 and 2 could be queried at once. This extra syntax remains optional. Following applies:

- When no syntax for channel addressing has been added, the command automatically addresses channel 1; it means in general, that channel 1 doesn't have to be addressed explicitly
- The syntax addition (or channel marker) @1 belongs to channel 1, the @2 to channel 2 etc.
- Channel addressing can be used on all commands supported by the addressed device/channel, even on global commands such as *IDN? which wouldn't cause a channel-specific reaction; in this case, the addressing is simply ignored
- Using a channel number that isn't supported by the addressed device, such as @5 on a three-channel model, is ignored; it means that it won't generate a communication error which could be queried by SYSTEM:ERROR?
- Returns from queries don't contain a channel marker, except for returns from error queries, because there multiple errors can be read at once from the error buffer, caused by different channels

Syntax rules of channel addressing for **set commands**, i. e. all <u>without</u> question mark at the end:

- **CURRENT␣5,␣(@2)** sets 5 A source current on channel 2
- **VOLTAGE␣12,␣(@1,3)** sets 12 V on channels 1 and 3
- **OUTP␣ON,␣(@1:3)** switches the DC terminals of channels 1 to 3 on

Syntax rules of channel addressing for **query commands**, i. e. all <u>with</u> question mark at the end:

- **MEASURE:ARR?␣(@3)** reads the three actual values of voltage, current and power from channel 3
- **MEASURE:ARR?␣(@1:3)** reads the three actual values of voltage, current and power from all three channels 1-3

5.2.4 Command concatenation

It's possible to concatenate up to 5 commands in one message. According to the SCPI standard, the commands then would be separated by a semicolon (;). The standard also allows to abbreviate commands with subsystems to make the string shorter, which is supported by the 20000 series parser, but not by the 10000 series parser. The logical solution would then be to always use complete commands, but with the 20000 series this would require to add a leading colon. Examples

Concatenated commands	Result
VOLT 80;CURR 20;POW 3kW	Works well all devices, because only command with 1 level are used.
VOLT:PROT?;CURR:PROT?	Would return the expected two values with a 10000 series device, while a 20000 series device would return only one value, plus error -113, because the leading colon is missing
VOLT:PROT?;:CURR:PROT?	Same as above, but with leading colon for the second command. Would return the expected two values from all devices
VOLT:PROT?;:LIM:HIGH?	Here, the :LIM and :PROT subsystems are on the same level. A 20000 series device would return the two expected values, while a 10000 series device in this case wouldn't return anything, but an error, because it misses a level with the 2nd command

The commands in the string are processed from left to right, so the order of commands is important to achieve correct results. When querying multiple values or parameters at once, the returned string is also in coupled format, with the queried returns separated by semicolons.

The device's internal buffer for SCPI strings is 256 bytes. Should the concatenated response string of a concatenated query exceed the buffer size, the device will not respond and instead generate error -225.

5.2.5 Termination character

Ethernet interfaces require to attach a termination character to the message, while others don't, such as USB. There the termination character is optional and used in order to maintain compatibility between several different interfaces in control softwares which use SCPI.

Supported termination character(s): **0xA** (LF, line feed)

The requirement for the termination character has been introduced in a specific firmware version. It may thus occur, that after an update of a device with a quite old firmware for which a software has been programmed, that this software cannot communicate anymore if it doesn't attach the termination character.

5.2.6 Communication and other errors

Errors in terms of SCPI are usually communication errors, but can also be extended by device specific alarms. According to the standard, devices using SCPI don't return errors immediately. They have to be queried from the device. The query can occur directly with the error command (see *5.4.5.3*) or by first reading the signal bit "err" from the STB register (see *"5.4.2. Status registers"*).

The error format is defined by the standard and is made of a string containing a number (the actual error code) and an explanatory text. Following errors strings can be generated by the device:

Error code / error text	Description
0,"No error"	No error
-100,"Command error"	Command unknown
-102,"Syntax error"	Command partially wrong
-108,"Parameter not allowed"	A command was sent with a parameter though the command doesn't use parameters
-200,"Execution error"	Command could not be executed
-201,"Invalid while in local"	Control command could not be executed, because device is in LOCAL mode
-220,"Parameter error"	Wrong parameter used
-221,"Settings conflict"	Command could not be executed because of the condition of the device being in the setup menu or isn't in remote mode yet
-222,"Data out of range"	Parameter could not be set because it exceeded a limit
-223,"Too much data"	Too many parameters per command or too many commands at once
-224,"Illegal parameter value"	A parameter not specified for the command has been sent
-225,"Out of memory"	A concatenated query has been sent whose concatenated response would exceed 256 characters (max. SCPI buffer length), such as possible with 5x *IDN?
-999,"Safety OVP"	Exception (device alarm), because no communication error: Safety OVP (only available with 60 V models of select series) has been triggered (see device manual). It requires to power cycle the device.

5.3 Examples for a first start

5.3.1 Ping or query device information

It's always recommended to ping a device first, in order to test if it responds at all. With SCPI, this is usually done by querying the identification string:

Protocol	Command
SCPI	*IDN?

As an immediate response, the device might send, for example:

Protocol	Response
SCPI	EA Elektro-Automatik GmbH&Co.KG, PSB 10080-340, 2023120002, V3.06 24.05.2023 V3.03 05.10.2022 V1.0.0

5.3.2 Switch to remote control or back to manual control

Before you can remotely control a device, you need to switch it to remote control by sending the dedicated command. Also see the SCPI command description below.

!	*The device will never switch to remote control mode automatically and can't be remotely controlled without being in this condition. Reading statuses and values is possible anytime.*

!	*The device would normally never exit remote control automatically, unless it's switched off or the AC supply is otherwise interrupted or the interface supervision timeout has run out and caused a CTO alarm. Remote control can be left by a certain command or by manual action on the HMI.*

Switching to remote control may be inhibited by several circumstances and is usually indicated by an error message:

- Condition **Local** is active (check the display or control panel on the front of your device), which will prevent any remote control
- The device is already remotely controlled by another interface
- The device in setup mode, means the user has accessed the setup menu and not left it yet

► How to switch a device to remote control:

1. If you are using SCPI command language, send a text command (the space is required):
   SYST:LOCK␣1 or SYST:LOCK␣ON

Leaving remote control can be done in two ways: using the dedicated command or by switching the device to "**Local**" condition. We will consider the first option, because this is about programming.

► How to exit remote control:

1. If you are using SCPI command language, send a text command (the space is required):
   SYST:LOCK␣0 or SYST:LOCK␣OFF

5.4 Command groups

Commands related to specific features of a device are grouped. Not all series feature the same number of commands. Every command below will indicate to which series it's compatible. Example:

10K	BT
✓	–

For the abbreviations in the table header see *"1.1.2. Validity"*, whereas

✓	means, the command or the command group (entirely or partly) is supported by the device series.
–	means, the command or the command group isn't supported by the device series.

5.4.1 Standard IEEE commands

In relation to the old interface standards GPIB and IEEE 488, some of the standard commands have been implemented. They are supported in all devices which feature SCPI command language.

5.4.1.1 *CLS

Clears the error queue, the status byte (STB) and all bits in the Event Status Register (ESR), except for bit 0.

5.4.1.2 *IDN?

Returns the device identification string, which contains following information, separated by commas:

1. Manufacturer
2. Model name
3. Serial number
4. Firmware versions of the three components KE, HMI and DR
5. User text (arbitrary user-definable text, as definable with SYST:CONFIG:USER:TEXT), if not empty

5.4.1.3 *RST

When sent, this will set the device to a defined state, except remote control is denied by the device:

1. Switch to remote control (same as SYST:LOCK␣1)
2. Set DC input/output to off
3. Clear alarm buffer
4. Clear status registers to default condition (QUEStionable Event, OPERation Event, STB)

5.4.1.4 *STB?

Reads the STatus Byte register. The signal run of the various device conditions and events is illustrated in the register model below. The STB bits in particular:

Bit 0: *sec_ques*, the second Questionable Status Register is active (one or several events have occurred)

Bit 1: not used

Bit 2: *err*, Error Queue --> one or several error in the error buffer. By reading the error buffer or sending *CLS it's flushed and the bit *err* is reset

Bit 3: *ques*, Questionable Status Register is active (one or several events have occurred)

Bit 4: not used

Bit 5: *esr*, Event Status Register is active (one or several events have occurred)

Bit 6: Master Summary Status, collective signal from the Service Request Enable register

Bit 7: *oper*, Operation Status Register is active (one or several events have occurred)

5.4.1.5 *ESR?

Reads the Event Status Register. This register holds signals that are related to SCPI command transfer and execution.

5.4.1.6 *ESE␣<NR1>

Reads with *ESE? or sets the Event Status Enable register, a signal filter for the ESR. The maximum filter value is determined by the supported bits (see register model below).

5.4.1.7 *SRE␣<NR1>

Reads with *SRE? or sets the Service Request Enable register.

5.4.2 Status registers

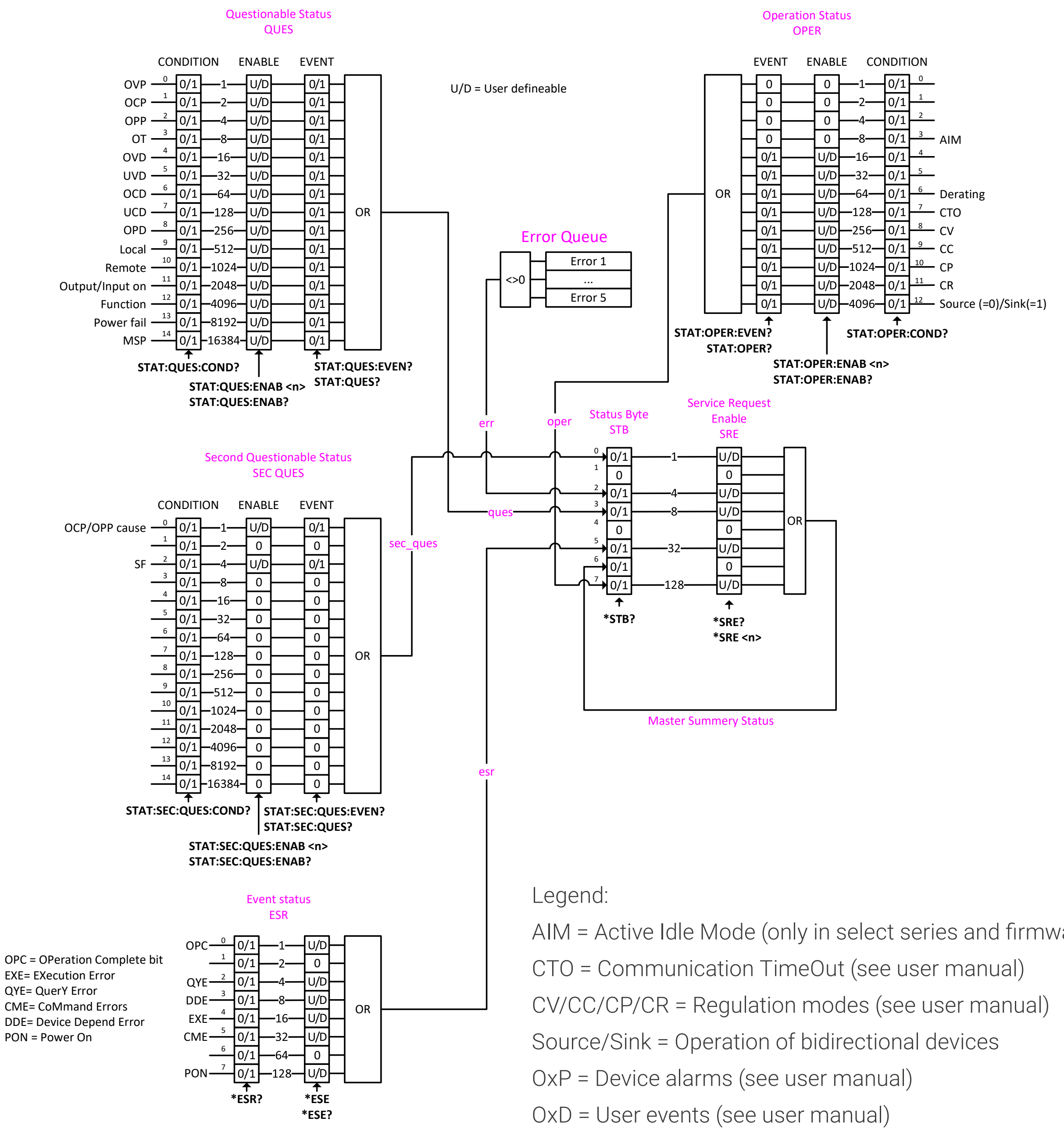
Most device conditions can't be read with dedicated SCPI commands, so they're grouped in status registers. They can be read arbitrarily, either direct or indirect. Indirect would be regular polling of the status byte (STB). It tells which linked status register (see model below) has recorded at least one event. The difference when reading the status registers directly would be, that the user would have to find out if a register has changed and which one. The bits in the status byte register will do that job for you. If they remain 0, nothing has happened.

Once a bit in the STB signalizes that there was an event recorded in QUEStionable (short: QUES) or OPERation (short: OPER) register, you could read the corresponding event or CONDition register of OPER and QUES, in order to find out which bits have changed. Disadvantage when reading the EVENT register: it only signals positive bit changes (0 -> 1), so that it would not signal when an alarm, for instance OVP, is gone already until the EVENT register is read again. The advantage is, on the other hand, that it allows the user to read and record alarms even after they're gone already.

By default, all :ENABle registers are 0 upon device start.

With multi-channel devices, such as those from BT 20000 Triple series, the registers exists three times in a device, one set for every channel. They're queried per channel as described below. However, they still contain global statuses, such as "Derating" which would then be equal for all channels.

Register model:



Legend:

AIM = Active Idle Mode (only in select series and firmwares)

CTO = Communication TimeOut (see user manual)

CV/CC/CP/CR = Regulation modes (see user manual)

Source/Sink = Operation of bidirectional devices

OxP = Device alarms (see user manual)

OxD = User events (see user manual)

MSP = Master-slave protection (see user manual)

The bits in the CONDition subregister of OPERation and QUEStionable are linked to the status bits of ModBus registers 505 and 511. The above shown overview is valid for a lot of series and not every series supports all bits shown, so there's a rule of thumb: if any status bit shown in the above register model isn't also listed in the ModBus register list for your particular device, then it's not supported by the device.

Device alarms like OVP are signaled in the subregisters :CONDition and :EVENT. They have to be acknowledged separately using commands SYST:ERR? or SYST:ERR:ALL?, which is considered as alarm acknowledgment and will clear the corresponding bit in :CONDITION, but only if the alarm condition isn't present anymore. Acknowledged alarms can later only be read from the device in form of an alarm counter. It's recommended to regularly poll alarms from the device and to query STAT:QUES? prior to SYST:ERR?.

Only with 60 V models: the additional alarm "Safety OVP" (SOVP, see device manual), isn't signaled in a different way, as a combination of alarm PF (Questionable Status, Bit 13) and alarm OVP (Questionable Status, Bit 0). Additionally, the unerasable error -999 is put into the error queue. SOVP can only be acknowledged by power-cycling the device.

5.4.2.1 STATus:QUEStionable?

Reads the Questionable status EVENT or CONDITION register. The device will return a 16 bit value, which represents device information as defined in the register model in *5.4.2*.

Query form 1:	STATus:QUEStionable:CONDition?
Query form 2:	STATus:QUEStionable:EVENt?
Query form 3:	STATus:QUEStionable?
Examples:	
STAT:QUES? --> 3072	Reads the event register. This value tells, that bits 10 and 11 are set and according to the register model this is interpreted as “remote control = active” and “DC input/output = on”.
STAT:QUES:COND?	Reads the condition register of the questionable status register. The value contains the current snapshot of a number of status bits.

5.4.2.2 STATus:QUEStionable:ENABle␣<NR1>

This command sets or reads the Enable register of the Questionable status register. The Enable register is a filter that enables all or single bits to signalise an event to the status byte STB. By default, all bits of the Enable register are set. In case you want to ignore certain bits, you just need to add the values of the remaining bits and send the value to the Enable register.

Query form:	STATus:QUEStionable:ENABle?
Value range:	Depends on the series. The maximum is the sum of all bits valid for the currently used device.
Example:	
STAT:QUES:ENAB␣3072	Sets the enable register of the questionable status registers to 3072 and enables the bits **OVP**, **OT**, **Remote** and **Input/Output on** for event reporting to STB.

5.4.2.3 STATus:OPERation?

Reads the Operation status EVENT or CONDITION register. The device will return a 16 bit value, which represents device information as defined in the register model in *5.4.2*.

Query form 1:	STATus:OPERation:CONDition?
Query form 2:	STATus:OPERation:EVENt?
Query form 2:	STATus:OPERation?
Examples:	
STAT:OPER? --> 256	Reads the operation register (identical to :EVENt?). A possible response would be a value of 256, which tells, that bit 8 is set and according to the register model bit 8 signalises, that „CV“ (constant voltage regulation) is active.
STAT:OPER:COND?	Reads the condition register of the operation status registers.

5.4.2.4 STATus:OPERation:ENABle␣<NR1>

Sets or reads the Enable register of the Questionable status register. The Enable register is a filter. It enables single or all bit of the condition registers to change the corresponding bit in the event register. This also impacts the summary bit in the status byte STB. By default, all bits of the Enable register are set to 1. If you want to use only some specific bits to be left through, just add their bit values (see register model) and send the total to the Enable register.

Query form:	STATus:OPERation:ENABle?
Value range:	Depends on the series. The maximum is the sum of all bits valid for the currently used device.
Example:	
STAT:OPER:ENAB␣1792	Sets the Enable register of the Operation register to value 1792 and enables bits „CV“, „CC“ and „CP“ for reporting events to the STB.

5.4.2.5 Further status registers

The 10000s series require additional alarm bits which led to the addition of a second questionable register which has these new alarms. See the register model above. There are also additional command for that new register which have the same use and function as the commands described in *5.4.2.1* to *5.4.2.4*. For details refer to these section. It also means, that so only the 10000s series support the extra commands:

STATus:SECond:QUEStionable?

STATus:SECond:QUEStionable:ENABle?

STATus:SECond:QUEStionable:ENABle␣<NR1>

STATus:SECond:QUEStionable:CONDition?

STATus:SECond:QUEStionable:EVENt?

5.4.3 Set value commands

All values which have dedicated commands are not only limited by the rated values of your particular device model, but additionally limited by adjustment limits, as definable in the setup menu or by additional commands in remote control. See below at 5.4.8.

5.4.3.1 [SOURce:]VOLTage˽<NR2>

Sets the voltage limit of the device or addressed channel (of multi-channel devices) within a certain range, which is either defined by adjustment limits ("Limits", where featured) or is 0...102% nominal value, or reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the voltage to the minimum or maximum adjustment limit.

Query form:	[SOURce:]VOLTage?
Value range:	<NR2> = 0...1.02 * rated voltage (according to technical specs)
Examples:	
VOLT 12	Absolute short form. Sets 12 V.
SOUR:VOLTAGE˽24.5V	Mixed form short/long, with unit. Sets 24.5 V, unless the voltage adjustment range has been limited otherwise.
SOURCE:VOLTAGE˽MIN	Sets the voltage to the defined minimum (U-min), usually 0 V

5.4.3.2 [SOURce:]CURRent˽<NR2>

Sets the current limit of the device or addressed channel (of multi-channel devices) within a certain range, which is either defined by adjustment limits ("Limits", where featured) or is 0...102% nominal value, or reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the current to the minimum or maximum adjustment limit. With bidirectional devices this command belongs to the source mode.

Query form:	[SOURce:]CURRent?
Value range:	<NR2> = 0...1.02 * nominal current (according to technical specs)
Example:	
CURR˽170	Absolute short form. Sets 170 A.
SOUR:CURRENT˽45.3A	Mixed form short/long, with unit. Sets 45.3 A, unless the adjustment range of the current has been limited otherwise.
SOURCE:CURRENT˽MAX	Sets the current to the defined maximum, which is either 102% of the rated current of the device or, if existing for the particular device, to the value of adjustment limit (I-max) (also see *5.4.8*).

5.4.3.3 [SOURce:]POWer˽<NR3>

Sets the power limit of the device or addressed channel (of multi-channel devices) within a certain range, which is either defined by adjustment limits ("Limits", where featured) or is 0...102% nominal value, or reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the power to the minimum or maximum adjustment limit. With bidirectional devices this command belongs to the source mode.

Query form:	[SOURce:]POWer?
Value range:	<NR3> = 0...1.02 * nominal power (according to technical specs)
Examples:	
POW˽3000	Absolute short form. Sets 3000 W, unless the power adjustment range has been limited otherwise.
SOUR:POWER˽3.5kW	Mixed form short/long, with unit and magnitude Kilo. Sets 3.5 kW or 3500 W, unless the adjustment range of the power has been limited otherwise.
SOURCE:POWER˽MIN	Sets the power to the defined minimum, which is 0 W, because there is no P-min.

5.4.3.4 [SOURce:]RESistance␣<NR2>

With electronic load devices, this command will set the input resistance value in Ohm within a range as defined in the technical specifications. Power supplies with internal resistance feature or bidirectional ones in source mode use this value to simulate an internal resistor in series to the output, where the output voltage differs from the adjusted value by an amount that calculates from the adjusted resistance value and actual output current. The way of setting the resistance value on both device types is identical. The adjustable range can be limited by an upper adjustment limit (R-max). Alternatively, parameters MIN or MAX can be used to instantly set the resistance to the adjustable minimum or maximum. With bidirectional devices this command belongs to the source mode of the device or the addressed channel (with multi-channel devices).

Query form:	[SOURce:]RESistance?
Value range:	<NR2> = Min. resistance...max. resistance, according to technical specs
Examples:	
RES?	Absolute short form. Queries the currently set resistance value.
SOUR:RESISTANCE␣10	Mixed form short/long. Sets 10 Ω.
SOURCE:RES␣MIN	Sets the resistance to the minimum defined for the particular device model.

5.4.3.5 SINK:CURRent␣<NR2>

This command is only available for bidirectional devices and sets the set value of current for the so-called sink mode of the device or the addressed channel (with multi-channel devices), which is separate from source mode (see *5.4.3.2*).

Contrary to the [SOURCE:]CURRent command, the main system SINK isn't optional, because the device could else not distinguish. The adjustment limits also apply, but for this separate set value there are also the separate limits, called "Sink: I-min" and "Sink: I-max", as adjustable on the HMI and by corresponding commands (see *5.4.8*). Alternatively, parameters MIN or MAX can be used to instantly set the sink current to the adjustable minimum or maximum.

Query form:	SINK:CURRent?
Value range:	<NR2> = I-min...I-max
Example:	
SINK:CURR␣120	Unless the adjustment limits restrict the setting, this will set the set value of current for the sink mode of a PSB to 120 A. This value can only become effective when the device changes into sink mode.

5.4.3.6 SINK:POWer␣<NR3>

This command is only available with bidirectional power supplies and sets the power set value for the so-called sink mode of the device or the addressed channel (with multi-channel devices), which is separate from the one of the source mode (see *5.4.3.3*).

Contrary to the [SOURCE:]POWer command, the main system SINK isn't optional, because the device could else not distinguish. The adjustment limits also apply, but for this separate set value there is also the separate limit "Sink: P-max", as adjustable on the HMI, as well as the corresponding command (see *5.4.8*). Alternatively, parameters MIN or MAX can be used to instantly set the power to the adjustable minimum or maximum.

Query form:	SINK:POWer?
Value range:	<NRf> = 0...P-max
Examples:	
SINK:POW␣4500	Unless the adjustment limit P-max restricts the setting, this will set the power for the sink mode to 4500 W. This value can only become effective after the device has switched to sink mode.
SINK:POWER␣MIN	Sets the power to 0 W.

5.4.3.7 SINK:RESistance␣<NR2>

This command is only available with bidirectional power supplies and sets the resistance set value for the so-called sink mode of the device or the addressed channel (with multi-channel devices), which is separate from the one of the source mode (see *5.4.3.4*).

Contrary to the [SOURCE:]RESistance command, the main system SINK isn't optional, because the device could else not distinguish. The adjustment limits also apply, but for this separate set value there is also the separate limit "Sink: R-max", as adjustable on the HMI, as well as the corresponding command (see *5.4.8*). Alternatively, parameters MIN or MAX can be used to instantly set the resistance to the adjustable minimum or maximum.

Query form:	SINK:RESistance?
Value range:	<NR2> = min.resistance...max. resistance (see technical specs) or R-max
Examples:	
SINK:RESISTANCE␣MIN	Sets the resistance set value for the sink mode to minimum as defined by the technical specifications, which varies from model to model. The ratings (or nominal values) can be queried from the device with further commands.
SINK:RESISTANCE␣5.2Ω	Sets the resistance set value for the sink mode to 5.2 Ohms, if the value is valid for the device or the addressed channel.

5.4.4 Measuring commands

Measuring commands return the last actual values which have been acquired by either measurement (U, I) or calculation (P, R). They represent the last situation on the DC terminal of the device or of the addressed channel of multi-channel devices. Actual values are acquired asynchronously to the measuring commands. It means, the value or values are not measured in the moment of query. Actual values are not necessarily identical to the corresponding set values. The device periodically acquires the actual values.

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

Queries the device to return the last measured voltage value on the DC terminal of the device or addressed channel in Volt.

Example:

MEAS:VOLT?	Absolute short form. Queries the actual voltage. A response, which should be instantly coming from the device, will return a value between 0% and max. 110% of rated device voltage, for example “43.50V”. The number of decimal places in the returned value will be identical to the value format on the device's display and varies from model to model.

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

Queries the device to return the last measured current value on the DC terminal of the device or addressed channel in Ampere.

With bidirectional devices, the returned actual value could be either from source or sink mode. Should the value be negative, i. e. have a leading minus, it belongs to sink mode.

Example:

MEASURE:CURRENT?	Queries the actual current only. A response, which should be instantly coming from the device, will return a value between 0% and max. 102% of nominal device current, for example “100.1A”. The number of decimal places in the returned value will be identical to the value format in the device display and varies from model to model.

5.4.4.3 MEASure:[SCALar:]POWer[:DC]?

Queries the device to return the last calculated power value on the DC terminal of the device or addressed channel in Watts.

With bidirectional devices, the returned actual value could be either from source or sink mode. Should the value be negative, i. e. have a leading minus, it belongs to sink mode.

Example:

MEAS:POW?	Absolute short form. Queries the consumed (e-load) or supplied power (PSU). A response, which should be instantly coming from the device, will return a value between 0% and max. 102% of nominal device power, for example “2534W”. No matter how the actual power format is on the device’s display, here it will always be returned in Watts.

5.4.4.4 MEASure:[SCALar:]ARRay?

Queries the device or the addressed channel (multi-channel devices) to return the last measured or calculated actual values of voltage, current and power (in that sequence) , separated by commas and with unit and probably magnitude. With multi-channel devices it's even possible to query all actual values from all channels at once, which would result in a longer string in which the values of channel 1 come first, then channel 2 etc.

With bidirectional devices, any of the returned actual values could be either from source or sink mode. Should any value be negative, i. e. have a leading minus, it belongs to sink mode.

Example:

MEAS:ARR?	Absolute short form. A response, which should be instantly coming from the device, will return three values between 0% and max. 125% of nominal device values, for example “12.5V, 33.3A, 420W”

5.4.5 Status commands

Status commands are used to alter the status of the device in terms of activating remote control or switching the DC terminal or to query the current status.

5.4.5.1 SYSTem:LOCK␣<B0>

This command is used to activate remote control of a device. Basically, remote control has to be activated first before you can send any command that changes device status or value. Once remote control has been activated via one of the digital interfaces, only that interface is in charge.

With multi-channel devices, the condition of remote control is considered the same as with a single channel device, because it's a global condition. It doesn't matter what channel is addressed when sending the remote on or off command, either all channels are in remote control or none.

The activation of remote control can be refused by the device due to several reasons. It's usually replied in form of a SCPI error which is put into the SCPI error buffer. This buffer can be read with the error command (see *"5.4.5.3. SYSTem:ERRor?"*).

Query form:	SYSTem:LOCK:OWNer?
Value range for set:	ON, OFF
Value range for query:	REMOTE, NONE, LOCAL
Examples:	
SYST:LOCK␣ON	Absolute short form. Requests the device to switch to remote control. The device then usually indicates activated remote control either by a LED or a status text in the display.
SYSTEM:LOCK:OWNER?	Queries the lock owner regarding remote control. This can be used to verify whether the device has accepted the request to switch to remote control or not. It can return three different statuses: REMOTE = Device is in remote control via any of the available interfaces NONE = Device isn't in remote control LOCAL = Device is in LOCAL condition, which denies or interrupts remote control. Usually actually manually on the device’s front panel

5.4.5.2 OUTPut␣<B0>

This command is used to switch the DC terminal of the device or the addressed channel (multi-channel devices) on or off or can be used to query the state.

Query form:	OUTPut?
Value range:	ON, OFF
Examples:	
OUTP␣ON	Absolute short form. Switches the DC terminal on if remote control is active.
OUTPUT?	Queries the condition of the DC terminal, which will be returned as ON or OFF.

5.4.5.3 SYSTem:ERRor?

This commands is used to read a single error or all errors from the device's internal SCPI error queue. This queue usually contains communication errors, such as about wrong syntax, excess values etc, but can also contain device alarm related errors, specifically one (SOVP). See section *"5.2.6. Communication and other errors"*. Other device alarms can be queried from the device by reading the status registers (see *"5.4.2. Status registers"*).

You can either chose to query the next error multiple times until it says "No error" or query all at once. After all errors have been read from the buffer, it will be purged.

Query form 1:	SYSTem:ERRor?	Queries the next error
Query form 2:	SYSTem:ERRor:NEXT?	Queries the next error
Query form 3:	SYSTem:ERRor:ALL?	Queries all errors in the buffer (up to 5)
Example:		
SYST:ERR?	Absolute short form. The device replies to this query with a string that first contains an error code (see error code list) and second an error description, for example: 0,"No error". This is returned every time no error is present or after all error have been returned. With multi-channel devices, the string would additionally contain a channel marker that tells which channel caused the error. Example: **-221,"Settings conflict;@1"**. The semicolon is used as separator.	
SYSTEM:ERROR:ALL?	This query will let the device return up to 5 concatenated errors in one string, each separated by comma+space.	

Querying errors with SYST:ERR? also clears bits related to device alarms in register QUEStionable (see "5.4.2. Status registers"), but only if the alarm condition is "gone". This is considered as acknowledgment Alarms that have been acknowledged this way can then not be read from the status register anymore.

5.4.6 Commands for protective features

All devices feature a set of device alarms, of which some are for self-protection, but others are for the protection of connected loads or sources. There is furthermore a supervision feature which can monitor DC terminal related values like voltage, current or power for exceeding adjustable limits and initiate user-definable actions like an acoustic alarm or shutdown of a DC terminal, either of the device or addressed channel. The configuration of the supervision can be done manually in the actual user profile or by remote commands.

5.4.6.1 [SOURce:]VOLTage:PROTection[:LEVel]␣<NR2>

This command is connected to an adjustable threshold related to OVP (overvoltage protection). The adjustable range is between 0 and 110% of the rated device or channel voltage. The threshold, when reached by the actual voltage value, would cause the device or the affected channel to switch off its DC terminal. It doesn't matter if the device or channel has generated the voltage exceeding this threshold or any outside source. When controlling a power supply, this feature serves to protect the connected load from overvoltage and thus damage, which might occur if the output voltage is accidentally adjusted to a dangerous level. Alternatively, parameters MIN or MAX can be used to instantly set the threshold to the adjustable minimum or maximum.

Query form:	[SOURce:]VOLTage:PROTection[:LEVel]?
Value range:	<NR2>: 0...1.1* rated voltage
Examples:	
VOLT:PROT␣88	Absolute short form. Sets the OVP threshold to 88 V. At a model with 80 V nominal voltage, this is 110% of the maximum voltage and also the maximum OVP value.

5.4.6.2 [SOURce:]CURRent:PROTection[:LEVel]␣<NR2>

This command is connected to an adjustable threshold related to OCP (overcurrent protection). The adjustable range is between 0 and 110% of the rated device or channel current of a multi-channel device. The threshold, when reached by the actual current, would cause the device or the affected channel to switch of its DC terminal. The threshold is only effective if it's adjusted to a lower or equal value than the DC current set value for sink or source mode, because else the device would just limit the current. Alternatively, parameters MIN or MAX can be used to instantly set the threshold to the adjustable minimum or maximum.

Query form:	[SOURce:]CURRent:PROTection[:LEVel]?
Value range:	<NR2>: 0...1.1 * rated current (of the device or channel)
Example:	
CURR:PROT␣100	Absolute short form. Sets the OCP threshold to 100 A.

5.4.6.3 [SOURce:]POWer:PROTection[:LEVel]␣<NR3>

This command is connected to the adjustable value “OPP” (overpower protection). The adjustable range is between 0 and 110% rated power of the device or channel. The threshold, when reached by the actual power, would cause the device or the affected channel to switch of its DC terminal The threshold is, however, only effective if it's adjusted to a lower or equal value than the DC power set value for sink or source mode, because else the device would just limit the power. Alternatively, parameters MIN or MAX can be used to instantly set the threshold to the adjustable minimum or maximum.

Query form:	[SOURce:]POWer:PROTection[:LEVel]?
Value range:	<NR2>: 0...1.1 * rated power (of the device or channel)
Example:	
POW:PROT␣3000	Absolute short form. Sets the OPP threshold to 3000 W.
POW:PROT␣1.5kW	Short form with unit and magnitude. Sets the OPP threshold to 1.5 kW.

5.4.6.4 SINK:CURRent:PROTection[:LEVel]␣<NR2>

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	✓	–

This command is only available with bidirectional power supplies and sets the so-called OCP threshold for the sink mode, which is separate from the OCP threshold of the source mode (see *5.4.6.2*).

Contrary to the command for the source mode, the main system SINK is here not optional, because the device could else not distinguish. No adjustment limits. Alternatively, parameters MIN or MAX can be used to instantly set the threshold to the adjustable minimum or maximum.

Query form:	SINK:CURRent:PROTection[:LEVel]?
Value range:	<NR2> = 0...1.1 * rated current (of the device or channel)
Examples:	
SINK:CURR:PROT␣MAX	Absolute short form. Sets the OCP threshold to 110% of the rated current

5.4.6.5 SINK:POWer:PROTection[:LEVel]␣<NR3>

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	✓	–

This command is only available with bidirectional power supplies and sets the so-called OPP threshold for the sink mode, which is separate from the OPP threshold of the source mode (see *5.4.6.2*).

Contrary to the command for the source mode, the main system SINK is here not optional, because the device could else not distinguish. No adjustment limits. Alternatively, parameters MIN or MAX can be used to instantly set the threshold to the adjustable minimum or maximum.

Query form:	SINK:POWer:PROTection[:LEVel]?
Value range:	<NR3> = 0...1.1 * rated power (of the device or channel)
Examples:	
SINK:POWER:PROT␣3000	Sets the OPP threshold to 5000 W, if the device or the addressed channel has at least 5000 W of rated power.

5.4.7 Commands for user-definable events

The commands below allow the remote configuration of user-definable events, also called supervision, where the device or channel (multi-channel) would monitor values related to voltage, current or power on the DC terminal.

5.4.7.1 Source mode related commands

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

Command	Description
SYSTem:CONFig:UVD[?]␣<NR2> **SYSTem:CONFig:UVD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event UVD, as configurable in the device's menu
SYSTem:CONFig:UCD[?]␣<NR2> **SYSTem:CONFig:UCD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event UCD, as configurable in the device's menu
SYSTem:CONFig:OVD[?]␣<NR2> **SYSTem:CONFig:OVD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event OVD, as configurable in the device's menu
SYSTem:CONFig:OCD[?]␣<NR2> **SYSTem:CONFig:OCD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event OCD as configurable in the device's menu
SYSTem:CONFig:OPD[?] <NR3> **SYSTem:CONFig:OPD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event OPD, as configurable in the device's menu

The **:ACTion** can have following parameters (also see the device's operation guide):

NONE = Event inactive, no supervision

SIGNAL = As soon as the event occurs, a status text is presented in the status field of the device display or a bit in the Questionable Register (STAT:QUES:EVEN?) is set (see *"5.4.2. Status registers"*). The bit indicates, that a specific event has occurred. This can be used to record the event.

WARNING = As soon as the event occurs, a warning pop-up is presented in the device display or a bit in the Questionable Register (STAT:QUES:EVEN?) is set (see *"5.4.2. Status registers"*). The bit indicates, that a specific event has occurred. This can be used to record the event.

ALARM = As soon as the event occurs, a warning pop-up is presented in the device display, as well as an acoustic alarm is initiated and the DC terminal of the device or affected channel is switched off. A bit in the Questionable Register (STAT:QUES:EVEN?) is also set (see *"5.4.2. Status registers"*). It indicates that a specific event has occurred. This can be used to record the event.

The action ALARM lets the device act similar to when device alarms occur. However, device alarm still have priority. It means, that if, for example, the values OVP and OVD would be equal and the input/output voltage reaches that level, the device would initiate an OV alarm rather than an OVD event.

5.4.7.2 Sink mode related commands

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

The below commands are only for the supervision (events) in the sink mode of bidirectional devices and use the additional subsystem **:SINK** for distinction.

Command	Description
SYSTem:SINK:CONFig:UCD[?]␣<NR2> **SYSTem:SINK:CONFig:UCD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event setting "Sink: UCD", as configurable in the device's menu
SYSTem:SINK:CONFig:OCD[?]␣<NR2> **SYSTem:SINK:CONFig:OCD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event setting "Sink: OCD" as configurable in the device's menu
SYSTem:SINK:CONFig:OPD[?]␣<NR3> **SYSTem:SINK:CONFig:OPD:ACTion[?]␣{NONE | SIGNal | WARNing | ALARm}**	Identical to event setting "Sink: OPD", as configurable in the device's menu

5.4.8 Commands for adjustment limits

Adjustment limits are additional, globally effective, adjustable limits for the set values U, I, P and R. The purpose is to narrow the standard 0...102% adjustment range of set values and to prevent, for example, to accidentally set an excess voltage. There is also the overvoltage protection (OVP), but it's generally better to block irregular set values in the first place.

In case a set value is sent to the device that would exceed an adjustment limit, no matter if too high or too low, the device will ignore it and put an error into the error queue. At the same time it's impossible to set the lower adjustment limit (:LOW) higher than the related set value or, vice versa, the upper adjustment limit.

These commands are connected to the "Limits" setting as you can adjust them in the setup menu of your device, where an HMI with display is featured. Also refer to the user manuals of the devices for details.

Command	BT	PS1	PSI1	PSB1	PSBE1	ELR1
[SOURce:]VOLTage:LIMit:LOW[?]␣<NR2> Identical to value U-min, as configurable at the device	✓	✓	✓	✓	✓	✓
[SOURce:]VOLTage:LIMit:HIGH[?]␣<NR2> Identical to value U-max, as configurable at the device	✓	✓	✓	✓	✓	✓
[SOURce:]CURRent:LIMit:LOW[?]␣<NR2> Identical to value I-min, as configurable at the device	✓	✓	✓	✓	✓	✓
[SOURce:]CURRent:LIMit:HIGH[?]␣<NR2> Identical to value I-max, as configurable at the device	✓	✓	✓	✓	✓	✓
[SOURce:]POWer:LIMit:HIGH[?]␣<NR3>[Unit Identical to value P-max, as configurable at the device	✓	✓	✓	✓	✓	✓
[SOURce:]RESistance:LIMit:HIGH[?]␣<NR2> Identical to value R-max, as configurable at the device	✓	—	✓	✓	✓	✓
SINK:CURRent:LIMit:LOW[?]␣<NR2> Identical to value "Sink: I-min", as configurable at the device	✓	—	—	✓	✓	—
SINK:CURRent:LIMit:HIGH[?]␣<NR2> Identical to value "Sink: I-max", as configurable at the device	✓	—	—	✓	✓	—
SINK:POWer:LIMit:HIGH[?]␣<NR3> Identical to value "Sink: P-max", as configurable at the device	✓	—	—	✓	✓	—
SINK:RESistance:LIMit:HIGH[?]␣<NR2> Identical to value "Sink: R-max", as configurable at the device	✓	—	—	✓	✓	—

5.4.9 Commands for master-slave/master-auxiliary operation

The commands, as listed below, are used to remotely configure and control the master-slave mode (short: MS), as it's named with 10000 series, respectively the master-auxiliary mode (short: MA) as named with 20000 series. The different naming also leads to different command sets. For details about MS/MA refer to the device's operation manual.

Configuration and control require a certain procedure, as depicted by the list of commands below. Alternatively, the entire configuration could also be done on the HMI of the devices, where one with display is featured.

5.4.9.1 Configuration commands for 10000 series devices

Command	Description
SYSTem:MS:ENABle␣{ON | OFF}	Enables (ON) or disables (OFF) master-slave (MS) mode
SYSTem:MS:ENABle?	Queries, whether the MS is enabled or not
SYSTem:MS:LINK␣{MASTER | SLAVE} **SYSTem:MS:LINK?**	Defines or queries the role of the device in the MS system: **MASTER** = Device will be master unit **SLAVE** = Device will be slave unit (this would also be returned when querying while MS isn't enabled)
SYSTem:MS:INITialisation	Starts the MS initialization with the given settings. Also refer to the device's operating manual. After a successful initialization, the MS mode can be controlled with further commands. To test if the init has been successful, the next command can be used:
SYSTem:MS:CONDition?	Queries the result of a former MS init. Possible return values: **INIT** = Init was successful **NO INIT** = Init was not successful An init is also successful if there is only a master. In order to find out whether you have a complete MS system available or not, you would have to query the number of initialized units from the master with command **SYST:MS:UNITS?** (see below). Any value other than 0 means, the MS system is initialized and available for control.
SYSTem:MS:UNITs?	Queries the number of units that have been initialized successfully. The number can differ from the expected value, if the master did not initialize one or multiple slaves due to any reason. If only the master has initialized itself, the command will return a 1. ❗ *Depending on the KE firmware installed on your device, the returned value may not include the master itself as an initialized device.*
SYSTem:MS:TERMination␣{ON | OFF} **SYSTem:MS:TERMination?**	Exception: only supported by the 10000s series. This command is used to activate/deactivate the digital switch for the termination resistor on the master-slave bus.
SYSTem:MS:BIAS␣{ON | OFF} **SYSTem:MS:BIAS?**	Exception: only supported by the 10000s series. This command is used to activate/deactivate the digital switch for the additional BIAS resistors on the master-slave bus. Note: The switch is automatically set to ON when the device is MS master.

5.4.9.2 Configuration commands for 20000 series devices

Command	Description
SYSTem:MA:ENABle␣{ON | OFF}	Enables (ON) or disables (OFF) master-auxiliary (MA) mode
SYSTem:MA:ENABle?	Queries, whether the MA is enabled or not
SYSTem:MA:LINK␣{MASTER | AUX} **SYSTem:MA:LINK?**	Defines or queries the role of the device in the MA system: **MASTER** = Device will be master unit **SLAVE** = Device will be aux unit (this would also be returned when querying while MA isn't enabled)
SYSTem:MA:INITialisation	Starts the MS initialisation with the given settings. Also refer to the devices's operating manual. After a successful initialisation, the MS mode can be controlled with further commands. To test if the init has been successful, the next command can be used:

Command	Description
SYSTem:MA:CONDition?	Queries the result of a former MA init. Possible return values: **INIT** = Init was successful **NO INIT** = Init was not successful An init is also successful if there is only a master. In order to find out whether you have a complete MA system available or not, you would have to query the number of initialized units from the master with command **SYST:MA:UNITS?** (see below). Any value other than 0 means, the MA system is initialized and available for control.
SYSTem:MA:UNITs?	Queries the number of units that have been initialized successfully. The number can differ from the expected value, if the master did not initialize all auxiliary units due to any reason. If only the master has initialized itself, the query would return a 1.

5.4.10 Commands for general queries

Here are commands listed that can be used to query other information from the device, which isn't used so often.

Command
SYSTem:NOMinal:VOLTage? Queries the nominal, i.e. rated input/output voltage of a single device or an initialized master-slave or master-auxiliary system.
SYSTem:NOMinal:CURRent? Queries the nominal, i.e. rated input/output current of a single device or an initialized master-slave or master-auxiliary system.
SYSTem:NOMinal:POWer? Queries the nominal, i.e. rated input/output power of a single device or an initialized master-slave or master-auxiliary system.
SYSTem:NOMinal:RESistance:MINimum? Queries the minimum internal resistance value of a single device or an initialized master-slave or master-auxiliary system. This value is usually not zero with electronic loads.
SYSTem:NOMinal:RESistance:MAXimum? Queries the maximum internal resistance value of a single device or an initialized master-slave or master-auxiliary system.
SYSTem:DEVice:CLASs? Queries the device class and returns a value which defines to what series the device belongs to. This is an easy way to distinguish different device series or types, like an electronic load from a power supply or battery charger. For a list of device classes see *"A1. Device classes"*
DIAGnostic:INFormation:DEVice:OTIMe? Operation counter. Queries the operation time, i. e. the time the device has been powered and running. The returned value is in hours.
DIAGnostic:INFormation:DEVice:ONTime? Operation counter. Queries the "on time", i. e. the time the device's or addressed channel's DC terminal has been switched on. The returned value is in hours.
DIAGnostic:INFormation:DEVice:OFFTime? Operation counter. Queries the "off time", i. e. the time the device's or addressed channel's DC terminal has been switched off. The returned value is in hours.
FETCh:AHOur? Operation counter. Queries the ampere hours the device or addressed channel has either supplied or consumed during "on time", depending of the device type. With bidirectional power supplies, the value is only connected to source mode. The returned value is in Ah.
FETCh:WHOur? Operation counter. Queries the watt hours the device or addressed channel has either supplied or consumed during "on time", depending of the device type. With bidirectional power supplies, the value is only connected to source mode. The returned value is in kWh.
FETCh:SINK:AHOur? Operation counter. This command is only available with bidirectional power supplies. It queries the ampere hours the device or addressed channel has either consumed in sink mode, during "on time". The returned value is in Ah.
FETCh:SINK:WHOur? Operation counter. This command is only available with bidirectional power supplies. It queries the watt hours the device or addressed channel has either consumed in sink mode, during "on time". The returned value is in kWh.

5.4.11 Commands for general device or channel configuration

The commands as listed below are used to modify settings of the device configuration. The settings can be part of the current user profile (see device's operating manual). Any modification on the configuration requires activated remote control. These settings are automatically stored.

With multi-channel devices, many of the settings are channel-specific and thus must be addressed as such or else the command would always affect channel 1. Those being channel-specific are marked as such in the table.

Command	BT	10K	Channel-specific
POWer:STAGe:AFTer:REMote␣{ AUTO | OFF } **POWer:STAGe:AFTer:REMote?** Defines, how the DC input/terminal of the device shall be after leaving remote control. This is connected to the device's menu setting **DC terminal->State after remote** on the HMI with display. **AUTO** = Last condition remains **OFF** = DC input/output will be switched off (default setting)	✓	✓	✓
SYSTem:CONFig:CONTroller:SPEed␣{FAST | SLOW | NORMal} **SYSTem:CONFig:CONTroller:SPEed?** Defines the voltage regulator/controller speed setting. Refer to the user manual of the device for more information about what this is and its effect.	✓	✓	✓
SYSTem:CONFig:INPut:RESTore[?]␣{AUTO | OFF} **SYSTem:CONFig:OUTPut:RESTore[?]␣{AUTO | OFF}** Defines the condition of DC input/terminal after the device is powered. This is connected to the device menu setting **DC terminal->State after power ON** on the HMI with display. **AUTO** = DC input/terminal will be restored to the condition it had when powering the device off the last time **OFF** = DC input/terminal will always be off after powering the device (default setting)	✓	✓	✓
SYSTem:CONFig:USER:TEXT␣<SRD> **SYSTem:CONFig:USER:TEXT?** Writes or queries a user-definable text of up to 40 ASCII characters permanently to the device. This string can be used to add custom information to the unit in order to distinguish it from other identical models, alternatively to the serial number. Multi-channel device don't have a user text per channel.	✓	✓	–
SYSTem:CONFig:ANALog:MONitor␣{DEFault | EL | PS | ELPS | PSEL | COMBination} **SYSTem:CONFig:ANALog:MONitor?** Configures the monitor pins 9 (VMON) and 10 (CMON) of the analog interface (where present) of bidirectional devices. **DEFault** = VMON signals actual voltage, CMON signals actual current of source or sink mode **EL** = Pin 10 only signals the actual current of sink mode **PS** = Pin 10 only signals the actual current of source mode **ELPS** = Pin 9 signal the actual current of sink mode, pin 10 the one of source mode **PSEL** = Inversion of **ELPS** setting **COMBination** = Pin 10 signals the current of source and sink mode, separating the signal range in two parts as -100%...0...100%. For more see device manual.	–	✓ (1	–
SYSTem:CONFig:ANALog:PIN6␣{OT | PF | ALL} **SYSTem:CONFig:ANALog:PIN6?** Defines what device alarms are signaled on pin 6. **OT** = Pin 6 only signals **O**ver**T**emperature **PF** = Pin 6 only signals **P**ower **F**ail **All** = Pin 6 signals both (default setting)	–	✓	–
SYSTem:CONFig:ANALog:PIN14␣{OVP | OCP | OPP | OVP/OCP | OVP/OPP | OCP/OPP | ALL} **SYSTem:CONFig:ANALog:PIN14?** Defines what device alarms are signaled on pin 14. There are options to signal the three device alarms **OVP**, **OCP** and **OPP** separately or as combination of two or signal all (logical OR)..	–	✓	–

1) Exception: Command available for bidirectional series PSB and PSBE only

Command	BT	10K	Channel-specific
SYSTem:CONFig:ANALog:PIN15␣{CONT | POW} **SYSTem:CONFig:ANALog:PIN15?** Defines what status is signaled on pin 15. **CONT** = Regulation mode CV (default setting) **POW** = DC terminal on/off	—	✓	—
SYSTem:CONFig:ANALog:REFerence␣{5 | 10} **SYSTem:CONFig:ANALog:REFerence?** Selects the voltage range for analog inputs and outputs of the analog interface. This has no effect on anything concerning digital remote control. **5** = 0...5 V range **10** = 0...10 V range (default setting)	—	✓	—
SYSTem:CONFig:ANALog:REMSb:LEVel␣{NORMal | INVerted} **SYSTem:CONFig:ANALog:REMSb:LEVel?** Determines how pin REM-SB of the analog interface (see device manual) shall be interpreted by the device: **NORMAL** = level and conditions as described in the manual (factory setting) **INVERTED** = level and conditions are interpreted as inverted	—	✓	—
SYSTem:CONFig:ANALog:REMSb:ACTion␣{OFF | AUTO} **SYSTem:CONFig:ANALog:REMSb:ACTion?** Determines the action that is caused by using pin REM-SB of the analog interface in connection with DC input/output of the device: **OFF** = pin can only be used to switch the DC input/output off (default setting) **AUTO** = pin can be used to switch off and on again, if the DC input/output was at least switched on once by pushbutton on the control panel or digital command	—	✓	—
SYSTem:CONFig:MODe␣{UIP | UIR} **SYSTem:CONFig:MODe?** Selects the regulation mode of the DC power stage of the device or the addressed channel between U/I/P (default setting) and U/I/R. By selecting U/I/R, resistance mode becomes effective with its resistance value (see commands **[SOURce:]RESistance** or **SINK:RESistance**). Activated U/I/R mode would be indicated in the display (where present) by the resistance value being shown either additionally to U, I and P or alternatively to P. In remote control, it would simply be queried.	✓	✓	✓
SYSTem:CONFig:SEMif47␣{DISable | ENAble} **SYSTem:CONFig:SEMif47?** Enables or diasbles a feature called SEMIF47 which is standard with units produced after approx. 04/2022. Also see the latest user manual or at least one issued after 04/2022 for details about this feature.	—	✓	—
SYSTem:COMMunicate:PROTocol:MODBus␣{ENABle | DISable} **SYSTem:COMMunicate:PROTocol:MODBus?** Enables or disables ModBus protocol on the device. This setting is stored. After disabling ModBus with this command, further ModBus messages are ignored, so that only SCPI commands are accepted. Only one of both protocols can be deactivated at the same time.	✓	✓	—
SYSTem:COMMunicate:TIMeout␣<NR1> **SYSTem:COMMunicate:TIMeout?** Defines a timeout in milliseconds (range: 5...65535, factory setting: 5) that can elapse between two consecutive bytes in a serial transfer (USB, RS232) before the device considers the message as interrupted and discards it. For details refer to section *3.6*.	✓	✓	—
SYSTem:COMMunicate:MONitoring:TIMeout␣<NR1> **SYSTem:COMMunicate:MONitoring:TIMeout?** Defines a timeout in seconds (range: 1...36000, factory setting: 5) for the so-called "interface monitoring" feature. Refer to the latest issue of the device manual for more information or contact support staff. In case the latest available manual for a particular series on our website still doesn't contain a paragraph about this feature, refer to another series' manual.	✓	✓	—

Command	BT	10K	Channel-specific
SYSTem:COMMunicate:MONitoring:ACTion␣{ON | OFF} **SYSTem:COMMunicate:MONitoring:ACTion?** Enables/disables the "interface monitoring" feature, also called "connection timeout" (short: CTO). This is an automatically saved setting. The actual monitoring via the timeout starts in the moment when a connection to the device via any digital interface is established. Also see the other command one row above. Once the time runs out, the described action is triggered and bit CTO in status register STAT:OPER:COND is set. **ON** = Enables the feature **OFF** = Disables the feature (default setting)	✓	✓	–
SYSTem:ALARm:ACTion:PFAil␣{ AUTO | OFF } **SYSTem:ALARm:ACTion:PFAil?** Defines, how the DC input/terminal of the device shall be after a power fail alarm has occurred and is gone before the device was powered down. The alarm itself would switch the DC terminal of the device or the affected channel temporarily off. This setting is connected to the device's menu setting **DC terminal->State after PF alarm** on the HMI with display. **AUTO** = The DC input/terminal condition before PF is restored **OFF** = The DC input/terminal remains switched off (default setting)	✓	✓	✓
SYSTem:ALARm:ACTion:OTEMperature␣{ AUTO | OFF } **SYSTem:ALARm:ACTion:OTEMperature?** Defines, how the DC input/terminal condition of the device or the affected channel shall be after the device/channel has recovered, i. e. cooled down after an overtemperature (OT) alarm. **AUTO** = The DC input/terminal condition before OT is restored (default setting) **OFF** = The DC input/terminal will remain switched off	✓	✓	✓

5.4.11.1 Commands related to digital interfaces

The 10000 series support replaceable Anybus interface modules of which some support sending SCPI command and could therefore be remotely configured using SCPI, either via USB port or even via the interface itself. The 20000 series have some digital interfaces built in and in total have less different interface types available, so that not all below listed commands are valid for the 20000s. Some commands are channel-specific for multi-channel devices and will be marked in the table below.

These settings are always saved automatically.

Command	Description	BT	10K	Channel specific
SYSTem:COMMunicate:INTerface:CODe?	Returns a value, representing a model code for the installed Anybus interface module: 5 = Profibus 9 = RS232 16 = CANopen 18 = ModBus TCP 1P 19 = Profinet/IO 1P 20 = Ethernet 1P 21 = Ethernet 2P 22 = ModBus TCP 2P 23 = Profinet/IO 2P 25 = CAN 26 = EtherCAT 255 = no module	—	✓	—
SYSTem:COMMunicate:INTerface:TYPe?	Queries the name of the installed Anybus interface module.	—	✓	—
SYSTem:COMMunicate:INTerface:SERial?	Queries the serial number of the installed Anybus interface module.	—	✓	—
SYSTem:COMMunicate:INTerface:ADDRess␣<NR1> **SYSTem:COMMunicate:INTerface:ADDRess?**	Sets the device address of the Profibus module IF-AB-PBUS or CANopen module IF-AB-CANO or queries it. Allowed range for Profibus: 1...125 Allowed range for CANopen: 1...127	—	✓	—
SYSTem:COMMunicate:PROFibus:ID?	Queries the Profibus ID of the device manufacturer.	—	✓	—
SYSTem:COMMunicate:PROFibus:FTAG␣<SRD> **SYSTem:COMMunicate:PROFibus:FTAG?**	Sets or queries the Profibus/Profinet specific „function tag“, a string of up to 32 characters	—	✓	—
SYSTem:COMMunicate:PROFibus:LTAG␣<SRD> **SYSTem:COMMunicate:PROFibus:LTAG?**	Sets or queries the Profibus/Profinet specific „location tag“, a string of up to 22 characters	—	✓	—
SYSTem:COMMunicate:PROFibus:DATe␣<SRD> **SYSTem:COMMunicate:PROFibus:DATe?**	Sets or queries the Profibus/Profinet specific „date tag“, a date/time string of up to 40 characters	—	✓	—
SYSTem:COMMunicate:PROFibus:DESCription␣<SRD> **SYSTem:COMMunicate:PROFibus:DESCription?**	Sets or queries the Profibus/Profinet specific „description“ tag, a string of up to 54 characters	—	✓	—
SYSTem:COMMunicate:PROFibus:NAMe␣<SRD> **SYSTem:COMMunicate:PROFibus:NAMe?**	Sets or queries the „station name”, a string of up to 200 characters	—	✓	—

Command	Description	BT	10K	Channel specific
SYSTem:COMMunicate:INTerface:BAUD␣<NR1> **SYSTem:COMMunicate:INTerface:BAUD?**	Queries or sets the bus speed, i.e. baud rate of a CANopen, CAN (not CAN FD) or RS232 interface. The device will only save the value. This means, with value 3 being saved and CANopen installed (10000 series) or selected (20000 series), it would result in 100 kbps and with the RS232 moduole installed (10000 series) it would result in 19200 Baud. **Value** / **CANopen** / **CAN** / **RS232**: 0 / 10 kbps / 10 kbps / 2400 Bd 1 / 20 kbps / 20 kbps / 4800 Bd 2 / 50 kbps / 50 kbps / 9600 Bd 3 / 100 kbps / 100 kbps / 19200 Bd 4 / 125 kbps / 125 kbps / 38400 Bd 5 / 250 kbps / 250 kbps / 57600 Bd 6 / 500 kbps / 500 kbps / 115200 Bd 7 / 800 kbps / 1 Mbps / - 8 / 1 Mbps / - / - 9 / Auto / - / -	✓	✓	—
SYSTem:COMMunicate:CAN:BROadcast␣<NR1> **SYSTem:COMMunicate:CAN:BROadcast?**	Sets the CAN broadcast ID for normal CAN communication. This ID is used to broadcast the message to multiple devices on a CAN channel at once and/or multiple channels of a multi-channel device which share the same broadcast ID. Allowed range: **0...2047** (11 bit) or **0...536870911** (29 bit)	✓	✓	—
SYSTem:COMMunicate:CAN:DLC␣{AUTO | FILL} **SYSTem:COMMunicate:CAN:DLC?**	CAN data length setting for response messages from the device. **AUTO** = the number of data bytes in a CAN message from the device (response) varies according to the used command/register (default) **FILL** = the number of data bytes in a CAN message is always 8, filled with zeros	✓	✓	—
SYSTem:COMMunicate:CAN:FORMat␣{BASE | EXT} **SYSTem:COMMunicate:CAN:FORMat?**	Selects the CAN ID format. **BASE** = 11 Bit (CAN 2.0A) (default after device reset) **EXT** = 29 Bit (CAN 2.0B)	✓	✓	—
SYSTem:COMMunicate:CAN:NODe␣<NR1> **SYSTem:COMMunicate:CAN:NODe?**	For normal CAN communication, this command sets the CAN base ID of a single device or of the addressed channel (multi-channel device). In later CAN communication with multi-channel devices, this ID is used to address a specific channel and has to be assigned carefully, always considering the three IDs for noncyclical communication per device or per channel. Allowed range: **0...2047** (11 bit) or **0...536870911** (29 bit)	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:NODe␣<NR1> **SYSTem:COMMunicate:CAN:READ:NODe?**	Same as with the Base ID, also either for a device or the addressed channel. It defines the Base ID for cyclic reading. Also see section *8.3.5* for information about cyclic reading. Allowed range: **0...2047** (11 bit) or **0...536870911** (29 bit)	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:ACTual␣<NR1> **SYSTem:COMMunicate:CAN:READ:ACTual?**	Defines the interval (in milliseconds) for the cyclic read of the device's or channel's actual values (U, I, P). Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓

Command	Description	BT	10K	Channel specific
SYSTem:COMMunicate:CAN:READ:ALIMits␣<NR1> **SYSTem:COMMunicate:CAN:READ:ALIMits?**	Defines the interval (in milliseconds) for the cyclic read of the device's or channel's adjustment limits for U and I (bidirectional series: of source mode). Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:BLIMits␣<NR1> **SYSTem:COMMunicate:CAN:READ:BLIMits?**	Defines the interval for the cyclic read of the device's or channel's adjustment limits for P and R (bidirectional series: of source mode). Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:CLIMits␣<NR1> **SYSTem:COMMunicate:CAN:READ:CLIMits?**	**Only with bidirectional series** Defines the interval for the cyclic read of the device's or channel's adjustment limits of I, P and R for sink mode. Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:SETS␣<NR1> **SYSTem:COMMunicate:CAN:READ:SETS?**	Defines the interval for the cyclic read of the device's or channel's set values (U, I, P, R). Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:BSETs␣<NR1> **SYSTem:COMMunicate:CAN:READ:BSETs?**	**Only with bidirectional series** Defines the interval for the cyclic read of the device's or channel's set values of I, P and R for sink mode. Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:READ:STAT␣<NR1> **SYSTem:COMMunicate:CAN:READ:STAT?**	Defines the interval for the cyclic read of the device's or channel's status. Also see section *8.3.5*. Allowed parameter range: **0** (=cyclic read is deactivated) or **20...5000**	✓	✓	✓
SYSTem:COMMunicate:CAN:SEND:NODe␣<NR1> **SYSTem:COMMunicate:CAN:SEND:NODe?**	Sets the CAN base ID for cyclic send, which is channel-specific with a multi-channel device. Also see section *8.3.2*. Allowed range: **0...2047** (11 bit) or **0...536870911** (29 bit)	✓	✓	✓
SYSTem:COMMunicate:CAN:TERMination {ON | OFF} **SYSTem:COMMunicate:CAN:TERMination?**	Switches the integrated bus termination resistor in the CA interface module or built-in CAN port (20000 series) **ON** or **OFF** (default setting)	✓	✓	—
SYSTem:COMMunicate:CAN:FD␣{ON | OFF} **SYSTem:COMMunicate:CAN:FD?**	Activates with **ON** or deactivates with **OFF** (default setting) the CAN FD mode for the CAN interface of 20000 series devices. With the activation, two more CAN FD related commands are unlocked. See below.	✓	—	—
SYSTem:COMMunicate:CAN:DATarate␣{0 | 1} **SYSTem:COMMunicate:CAN:DATarate?**	After the activation of the CAN FD mode (see above at **:CAN:FD**), this command would select one out of two available CAN FD specific data baud rates: **0** = 500 KBit / 2 MBit **1** = 500 KBit / 5 MBit	✓	—	—
SYSTem:COMMunicate:CAN:BRS␣{ON | OFF} **SYSTem:COMMunicate:CAN:BRS?**	After the activation of mode CAN FD (see above at **:CAN:FD**), this command can be used to activate the so-called bit rate switching (BRS) mode for CAN FD with **ON**. This mode is deactivated (**OFF**)by default.	✓	—	—

5.4.11.2 Commands for Ethernet interface settings

The commands below are related to any Ethernet interface port, no matter if built-in or pluggable module. Some commands are only supported when an Anybus interface module (10000 series) is installed.

The so-called 10000s series even support two Ethernet interfaces. The built-in Ethernet port is here considered as the secondary port, no matter if an Anybus Ethernet module is installed or not. Hence for the 10000s series it's required to switch the access to the settings of the secondary port before reading/writing parameters. See the extra command below.

Overview of the Ethernet port availability:

	All 10000s series	All BT 20000 series
Primary port	Anybus (optional)	Rigidly built-in
Secondary port	Rigidly built-in	-

Command	Description	BT	10K
SYSTem:COMMunicate:LAN:1SPEed[?]␣{AUTO | 10HALF | 10FULL | 100HALF | 100FULL **SYSTem:COMMunicate:LAN:2SPEed[?]␣{AUTO | 10HALF | 10FULL | 100HALF | 100FULL**	Only for Anybus standard Ethernet modules and ModBus TCP modules: Sets the communication speed of the network P1 (belongs to **:1SPEED**) or (P2, belongs to **:2SPEED**), where featured: **AUTO** = Auto negotiation **10HALF** = 10MBit, half duplex **10FULL** = 10MBit, full duplex **100HALF** = 100MBit, half duplex **100FULL** = 100MBit, full duplex	—	✓
SYSTem:COMMunicate:LAN:ADDRess[?]␣<SRD>	Queries or sets the IP address of the selected Ethernet interface. When setting the IP, the string has to be in the typical IP format like this: 192.168.0.2	✓	✓
SYSTem:COMMunicate:LAN:CONTrol[?]␣<NR1>	Queries or sets the TCP port (0...65535) of the selected Ethernet interface. Default is **5025**, used for ModBus RTU or SCPI communication. Devices supporting ModBus TCP have port 502 activated and reserved by default, so 502 is illegal to be set with this command.	✓	✓
SYSTem:COMMunicate:LAN:DHCP[?]␣{ON | OFF}	Activates (=ON) or deactivates (=OFF) the DHCP functionality for the selected Ethernet interface. Default is OFF, so the IP as set with :ADDR command above is used.	✓	✓
SYSTem:COMMunicate:LAN:DNS1[?]␣<SRD>	Queries or sets the network address of the first (DNS1) domain name server	✓	✓
SYSTem:COMMunicate:LAN:DNS2[?]␣<SRD>	Queries or sets the network address of the second (DNS2) domain name server	—	✓
SYSTem:COMMunicate:LAN:DOMain[?]␣<SRD>	Queries or sets the domain name (refer to network terminology for details). This is a simple ASCII string of up to 54 characters. The domain can be used to select and access a particular device in the network without knowing the IP address.	✓	✓
SYSTem:COMMunicate:LAN:GATeway[?] <SRD>	Queries or sets the gateway address of the selected Ethernet interface. Format is the same as with the IP. This address is often not used and can be kept at its default setting.	✓	✓
SYSTem:COMMunicate:LAN:HOSTname[?]␣<SRD>	Queries or set the host name (refer to network terminology for details). This is a simple ASCII string of up to 54 characters.	✓	✓
SYSTem:COMMunicate:LAN:KEEPalive[?]␣{ON | OFF}	Enables/disables the so-called "TCP keep-alive" for the selected Ethernet interface. Also see *"3.7. Connection timeout"*. Default setting: OFF	✓	✓
SYSTem:COMMunicate:LAN:MAC?	Queries the MAC of the selected Ethernet interface, when physically present.	✓	✓

Command	Description	BT	10K
SYSTem:COMMunicate:LAN:SMASk[?]␣<SRD>	Queries or sets the subnet mask of the selected Ethernet interface. Format is the same as with the IP.	✓	✓
SYSTem:COMMunicate:LAN:TIMeout[?]␣<NR1>	Defines a socket connection timeout for the selected Ethernet interface in seconds. Also see *"3.7. Connection timeout"*. Setting this to 0 disables the timeout. Adjustment range: **0** or **5**...**65535**. Default setting: **5**	✓	✓

The 10000s series require to select between the primary Ethernet port (in the slot, Anybus module) and the secondary Ethernet port (built-in RJ45) before accessing Ethernet related settings via SCPI commands. After powering the device, the primary interface is selected by default, no matter if a module is plugged or not.

Command	Description	BT	10K
SYSTem:COMMunicate:LAN:INDex␣{1 | 2} **SYSTem:COMMunicate:LAN:INDex?**	Temporary switch between the secondary interface (=**2**) and the primary interface (=**1**, default after power-up).	—	✓

5.4.12 Commands for remote control of the function generator

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

- *Sequence data or table data, which you can write the below listed commands, aren't stored in the device*
- *With multi-channel devices, every channel features a function generator which is considered as completely separated!*

The function generator (short: FG) is a complex part of the whole control options of the device. It can be remotely configured and controlled by a set of SCPI commands. When operating the function generator on the control panel of the device, it requires a certain procedure of setup before getting to the actual starting point. The single commands can't enforce that procedure, so it's up to the user to use them in the correct order. What to do:

1) Select the type of generator

Before the first use, the function generator requires to be configured at least once after the device has been powered. The first step is to select the type of function generator, causing further steps to depend on your selection. There are two types available: **XY** and **arbitrary**.

The **XY** function generator is only a memory for a table with 4096 values which represent 0-125% actual values of the rated voltage or current of the device which actually, in case of the current, can only use the effective range of 0-102% and in case of voltage the voltage only 0-110% (OVP limit). Bidirectional devices feature two tables, one for source mode (generator mode: **IUPS**) and one for sink mode (generator mode: **IUEL**). The mode **IU** combines both of these modes into one, so that in this mode a bidirectional device can switch between the two quadrants while having different tables loaded source and sink mode. The switchover follows the general way, like in other modes of operation, by determining the difference between voltage set value and actual voltage. Sink mode generally only works if the voltage on the DC terminal is higher than the voltage set value.

The **arbitrary** generator is used to generated standard waves like sine, rectangle, triangle or custom ones.

2) Configure the function generator (part 1)

As a second step, when using the arbitrary generator, it requires to first assign the function to voltage (U) or current (I). After that you would define the range of sequence points to run through. This won't be set up automatically when filling a certain number of sequence points with data. The advantage is that a bigger number of sequence points could be loaded and only a portion would be set up to run through, but with n repetitions and after that, another portion could be used with a different number of repetitions.

In case the XY generator is used, the second step would be to select what sort of XY curve it shall run. Depending on that selection, values written to the table memory will be interpreted and checked for plausibility accordingly.

3) Configure the function generator (part 2)

The last step is to fill the function generator with data. With the arbitrary generator this is done by setting up X out of 99 possible sequence points. The number of effective points to run is variable, but at least 1.

The XY generator is filled with 1-4096 values per table.

XY data and sequence data, once completely loaded, require to be submitted by a dedicated command which requires to wait some time before proceeding. We recommend to wait at least 2 seconds after submitting the data and before the next command.

4) Use the function generator

After step 3, the function generator is completely configured and can be started.

Switching to a different function generator mode requires to first leave function generator mode by sending [SOURce:]FUNCtion:GENerator:SELect␣NONE. Only after that, the device will permit to select a different mode. Previously loaded table or sequence point data will be erased when leaving function generator mode.

5.4.12.1 XY generator: Mode selection and set up

The **XY** generator configuration of a device from a bidirectional series device is more complicated than with other series, because due to the nature of a bidirectional device with its sink and source modes, it has two data tables. **Table 1 is assigned to source mode, table 2 to sink mode.** Depending on the selected mode, either one or the other or both tables are used, which also requires to distinguish the commands to use. Mode comparison and overview:

XY mode	Description	Operation mode	Uses table(s)	Remarks	BT	PSI1	PSB1	ELR1
IU	IU curve	Source or sink	1 and 2	Active quadrant can be switched by varying the voltage set value	✓	—	✓	—
IUEL	IU curve for sink mode	Sink	2	Electronic loads naturally run in sink mode	✓	—	✓	✓
IUPS	IU curve for source mode	Source	1	Non-bidirectional power supplies naturally run in source mode	✓	✓	✓	—
PV	Simple PV, same as PVA	Source	1		✓	✓	✓	—
PVA	PV curve A	Source	1	Curve A is the default curve which is also used by both, the simple and extended PV functions. Curve B can be used alternatively.	✓	✓	✓	—
PVB	PV curve B	Source	2		✓	—	✓	—
FC	FC curve	Source	1	FC curves are actually UI curves which are loaded for the IU mode, so the table data either has to be transposed before or is directly calculated as current values by the FC formula.	✓	—	✓	—

Command	Description
[SOURce:]FUNCtion:GENerator:SELect␣{FC | IUPS | IUEL | UI | PV | PVA | PVB | NONE} **[SOURce:]FUNCtion:GENerator:SELect?**	Selects the run mode of XY function generator mode according to the table above. For the **extended EN 50530 PV** curve commands refer to section *5.4.15.* **NONE** = Exit function generator, also used before switching to another FG mode

All series that feature the XY generator offer to switch between 6 modes of the XY generator. Those modes not supported by the particular device type, see the table above, would simply not work when configured and run.

After the XY mode has been selected, table data can be loaded. Refer to section *5.4.12.* and also to the user manual of your device for further details about the XY function generator and how it works. According to the distinction of modes as shown above, the table assignment leads to the use of table assigned commands for the 1st and 2nd table:

Commands for XY table 1	Description
[SOURce:]FUNCtion:GENerator:XY:LEVel␣<NR1>	**1.** Selects a table entry (range: 0...4095) for writing or returns the currently selected entry number.
[SOURce:]FUNCtion:GENerator:XY:DATa␣<NRf> **[SOURce:]FUNCtion:GENerator:XY:DATa?**	**2.** Writes a value to the previously with **:LEVel** selected table entry or returns the value.
[SOURce:]FUNCtion:GENerator:XY:SUBMit␣FIRSt	**3.** Submits the data which has been written so far for the 1st table. After submitting the data the function can be started

Commands for XY table 2	Description
[SOURce:]FUNCtion:GENerator:XY:SECond:LEVel␣<NR1> **[SOURce:]FUNCtion:GENerator:XY:SECond:LEVel?**	**1.** Selects a table entry (range: 0...4095) for writing or returns the currently selected entry number.
[SOURce:]FUNCtion:GENerator:XY:SECond:DATa␣<NR2> **[SOURce:]FUNCtion:GENerator:XY:SECond:DATa?**	**2.** Writes a value to the previously with **:LEVel** selected table entry or returns the value.
[SOURce:]FUNCtion:GENerator:XY:SUBMit␣SECond	**3.** Submits the data which has been written so far for the 2nd table. After submitting the data the function can be started

5.4.12.2 Arbitrary generator: Mode selection and configuration

Command	Description
[SOURce:]FUNCtion:GENerator:SELect␣{VOLTage | CURRent | NONE} **[SOURce:]FUNCtion:GENerator:SELect?**	Select the function generator mode: **VOLTage** = Arbitrary generator for U **CURRent** = Arbitrary generator for I **NONE** = Exit function generator
[SOURce:]FUNCtion:GENerator:WAVe:STARt␣<NR1> **[SOURce:]FUNCtion:GENerator:WAVe:STARt?**	Defines the start sequence point (range: 1...99) or queries the last setting. If only one sequence point is used, then it must be **:END** = **:STARt**.
[SOURce:]FUNCtion:GENerator:WAVe:END␣<NR1> **[SOURce:]FUNCtion:GENerator:WAVe:END?**	Defines the end sequence point (range: 1...99) or queries the last setting.
[SOURce:]FUNCtion:GENerator:WAVe:NUMBer␣<NR1> **[SOURce:]FUNCtion:GENerator:WAVe:NUMBer?**	Defines, how often the sequence block defined by **:STARt** and **:END** is run through, or queries the last setting. Range: **0** (infinite cycles) or **1**...**999**

5.4.12.3 Arbitrary generator: Load sequence point data

Sequence point data should only be sent to the device after it was switched to function generator mode, which also sets the assignment of the arbitrary generator to U or I.

A function can consist of 1 to 99 sequence points, where one sequence point can be a complete function (sine) or just a part of it (ramp, rectangle). When started, the function generator will execute the sequence points from start sequence point to end sequence point, as defined by the user. With every point being variable, the resulting function can be quite complex. The sequence point data is loaded into the device with three commands and in a specific order like this:

Command	Description
[SOURce:]FUNCtion:GENerator:WAVe:LEVel␣<NR1> **[SOURce:]FUNCtion:GENerator:WAVe:LEVel?**	**1.** Selects a sequence point (range: 1...99) to write or queries the currently selected sequence point number
[SOURce:]FUNCtion:GENerator:WAVe:INDex␣<NR1> **[SOURce:]FUNCtion:GENerator:WAVe:INDex?**	**2.** For the selected sequence point, a set of parameters can be configured. This command selects the parameter between 0 and 7 with **INDex**. The next command **(:DATa)** is then used to write a specific value. The indexes are explained below. This can also be used to query the value of the currently selected index.
[SOURce:]FUNCtion:GENerator:WAVe:DATa␣<NR2> **[SOURce:]FUNCtion:GENerator:WAVe:DATa?**	**3.** This will write a value, for example a frequency, to the previously selected parameter, as part of the sequence point setup. This can also be used to query the last value.
[SOURce:]FUNCtion:GENerator:WAVe:SUBMit	**4**. Submits all data. Without sending this command, the FG wouldn't use the programmed data, but any previous data that still resides in the internal memory.

When manually adjusting parameters for the arbitrary function generator on the device's control panel (HMI), they are limiting each other so the resulting signal will work as expected. In remote control, the plausibility of values is also checked, but this is only done after the data has been submitted. The check result may lead to an error which should be queried from the device in order to proceed or to handle a value error.

For example, index 0 is connected to index 5, as the DC value is the base line of the AC amplitude. It means, if you would want to achieve a sine wave with 5 A amplitude on the DC input current of an electronic load, the base line of the resulting sine wave has to be at minimum 5 A, else the negative wave would be clipped at 0. Indexes 5 and 6 are adjustable DC offsets which move the AC wave's base line on the Y axis. So the values in indexes 5 and 6 should at least be as high as index 0 or 1 (whichever is bigger), but they can also be higher. See figures below or see the user manual of the device, in the section for the function generator where is has more XY diagrams of the waves.

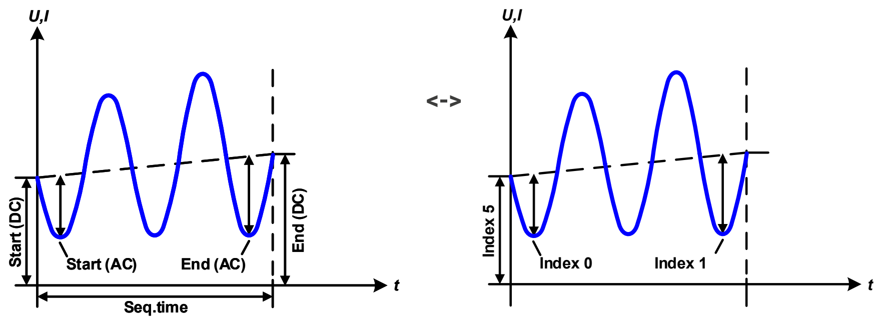
In relation to the adjustments for a function as they can be done on the device's front panel, following indexes are selectable with **:INDex** and readable/writable with **:DATa**:

Index	Parameter	Data type	Unit	Value range	Note
0	Start value (amplitude)	Float	A, V	0...Nominal value (U or I)	For AC part only
1	End value (amplitude)	Float	A, V	0...Nominal value (U or I)	For AC part only
2	Start frequency in Hz	Integer	Hz	0...10000	For AC part only
3	End frequency in Hz	Integer	Hz	0...10000	For AC part only
4	Start angle in °	Integer	-	0...359	For AC part only

Index	Parameter	Data type	Unit	Value range	Note
5	Start level (offset)	Float	A, V	0...Nominal value (U or I) Bidirectional series only: -Nom. value of I...+ Nom. value of I	For AC and DC part
6	End level (offset)	Float	A, V	0...Nominal value (U or I) Bidirectional series only: -Nom. value of I...+ Nom. value of I	For AC and DC part
7	Sequence time	Float	s	0.0001...36000	

In the user manual of your 10000 series device it might still describe a minimum slope for changes on the DC and AC part of a sequence point. The requirement of a minimum slope isn't valid for a 20000 series device and is also invalid for a 10000 series devices that have firmware KE 3.02 or higher installed. It means, for older firmware installations an update can remove the requirement and allows for ramps longer than the described 1379 seconds.

Parameter assignment illustrated by an example curve:



5.4.12.4 XY generator: Control

After the configuration of the XY generator and after all necessary table data has been loaded it can be started by simply switching the DC output/input of the device on. This is the actual function run and is only stops due to a device alarm or abortion by the user.

5.4.12.5 Arbitrary generator: Control

Contrary to the XY generator where the curve is immediately active when switching the DC terminal/input, the arbitrary generator requires run control by command. The run can't be paused. It means, once the function is stopped, no matter by what reason, the next start will run from the beginning, from the first sequence in use.

Command	Description
[SOURce:]FUNCtion:GENerator:WAVe:STATe␣{RUN | STOP} **[SOURce:]FUNCtion:GENerator:WAVe:STATe?**	Starts/stops the arbitrary function generator or queries the state

5.4.12.6 Special function "Simple PV" (photovoltaics)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	–	✓

For the extended PV function EN 50530 refer to section "5.4.15. Commands for the extended PV simulation".

The photovoltaics function (PV), as available in certain power supply series, is based on the XY function generator. In remote control it's required to load a complete, precalculated value table into the device, contrary to the more comfortable way on the HMI where you set up 4 simulation related parameters and the device would then calculate a table and use it. In order to have table data calculated and to load it into the device via an interface, you have two options:

1. Use external tools and save as CSV file which also allows for loading it later from any USB stick or let it be calculated in a custom software and directly transfer it to the device.

2. Use the HMI, enter four PV simulation related parameters, let the device calculate and load the table, then go back in the menu to find an option to save the table data to USB stick for later use, like loading its content via any interface into the device.

During the function run the irradiation value, which simulates different light situations, can be adjusted. The irradiation impacts the current in the maximum power point as a factor.

Command	Description
[SOURce:]IRRadiation␣<NR1> **[SOURce:]IRRadiation?**	Adjusts or queries the irradiation value during the solar panel simulation in a range of 0...100 per cent, which affects the DC current and shifts the MPP

Given that the device is already in remote control and the XY table for the PV function is already calculated and ready to be loaded, following procedure is defined:

► How to setup the device for the simple PV function

No.	Command	Description
1	FUNC:GEN:SEL␣PVA	Selects PV mode for the XY generator. By sending this command the device will switch to function generator mode.
2	FUNC:GEN:XY:LEV␣0	Subsequently write all XY table values for the PV curve into the device. The table can have up to 4096 values, corresponding to a measurable range of 0...125% of the rated voltage. Since a PV runs in source mode, the device can never generate more than 102% of rated voltage, so it makes sense to write less values, specifically 4096/125*102=3342.
3	FUNC:GEN:XY:DAT␣<NR2>	
...		
6684	FUNC:GEN:XY:3342	
6685	FUNC:GEN:XY:DAT␣<NR2>	
6686	FUNC:GEN:XY:SUBM	Submit the written data

After submitting all data, it's necessary to wait some time (≈1 s) before starting the simulation. Optionally and if not already done, set additional global limits like voltage and power, either to maximum or any other value that doesn't interfere the simulation. The voltage here should be set to the open circuit voltage (Uoc) of the simulated panel, alternatively to maximum:

No.	Command	Description
6687	VOLT␣MAX	Set voltage to max. Alternatively, the voltage could also be set to the Uoc of the simulated panel for a more correct simulation
6688	POW␣MAX	Set power to maximum, so constant power regulation won't interfere

After everything is set, you can run the function and control the simulation and irradiation. The irradiation then acts as a factor that is multiplied to the current value that is read from the table, so changing this value moves the power point. For an illustration of the PV curve, refer to your device's user manual.

► How to control the device during the PV function run

No.	Command	Description
6689	OUTP␣ON	Switch the DC terminal of your device on to start the function
6690	IRR␣85	Set irradiation to 85%, for example
6691	OUTP␣OFF	Switch the DC terminal of your device off to make the function stop
6692	FUNC:GEN:SEL␣NONE	Parameter **NONE** leaves the function generator mode

5.4.12.7 Special function: FC (fuel cell)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	–	✓

The fuel cell function (FC) is based on the XY function generator, as available in certain power supply series. In remote control it's only possible to load a complete, precalculated table with X values into the device. Basically, an FC curve is an UI curve. Since the 10000 series and 20000 series devices don't support UI curves, it requires a different approach. For FC simulation, the XY generator has a direct FC mode, which internally is based on IUPS mode. It requires to load current values into the table .

Loading table data into the device in remote control is contrary to the more comfortable way on the HMI where you define 4 points on the curve by entering values and the device would then calculate and use the table. But the HMI method can be used to save the calculated table data for use in remote control. In order to have the table data calculated there are two options:

1. Calculate it with external tools, such as MS Excel, and then load all data into the device.

2. Use the HMI, enter the four points on the curve, let the device calculate and load the table, then leave FG mode and go back to the menu, where is it would then show an option to save the table data to USB stick. The file can be loaded and the data in it transferred to the device as in option 1. Alternative: read the table data directly from the device via any interface and save as file.

The 2nd method is recommend for use with devices where it requires transposition from UI to IU, such as the 10000 and 20000 series, as there is no official formula available to translate an FC-UI table into current values (IU table).

5.4.13 Commands for remote control of the MPP tracking feature

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	—	✓	—	—

Maximum Power Point (MPP) tracking is something inverters for solar panels use. The MPP tracking, as available with electronic loads or bidirectional devices in sink mode, emulates the tracking behavior of such inverters. The feature itself and its modes and settings are described in the device manual. Below only the corresponding commands for remote setup and control are explained.

5.4.13.1 Configuration of the MPP tracking

The configuration is done with a few "indexes", representing different subfeatures, and a :**DATa** command:

Command	Description
[SOURce:]FUNCtion:GENerator:MPP:INDex␣0 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣{NONE | MPP1 | MPP2 | MPP3 | MPP4}**	Index 0: MPP tracking mode selection **MPP1**: MPP tracking mode 1 (Find MPP) **MPP2**: MPP tracking mode 2 (Track) **MPP3**: MPP tracking mode 3 (Fast track) **MPP4**: MPP tracking mode 4 (User curve) **NONE**: Deactivate MPPT
[SOURce:]FUNCtion:GENerator:MPP:INDex␣1 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 1: Sets the open circuit voltage (U_{OC}) Range: 0 - U_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣2 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 2: Sets the short-circuit current (I_{SC}) Range: 0 - I_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣3 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 3: Sets the voltage in the MPP (U_{MPP}) for fast track mode 3 Range: 0 - U_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣4 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 4: Sets the current in the MPP (I_{MPP}) for fast track mode 3 Range: 0 - I_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣5 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 5: Sets the power in the MPP (P_{MPP}, most significant regulation value) for fast track mode 3 Range: 0 - P_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣6 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 6: Sets ΔP (in Watts), a difference to the MPP above which the tracker starts to find the MPP again (only used with mode 2 and mode 3) The allowed range is defined as 0-50W for some series, for others it's 0 - P_{NOM}.
[SOURce:]FUNCtion:GENerator:MPP:INDex␣8 **[SOURce:]FUNCtion:GENerator:MPP:LEVel[?]␣<NR1>** **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR2>**	Index 8: Sets voltage values for mode 4 Select the value to set (1...100) Range: 0 - U_{NOM}
[SOURce:]FUNCtion:GENerator:MPP:INDex␣10 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR1>**	Index 10: Sets the regulation interval in milliseconds for mode 4 stepping or for next tracking action in the other mode (this parameter is only available in remote control; for manual control it's set to minimum) Range: 5...60000
[SOURce:]FUNCtion:GENerator:MPP:INDex␣11 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR1>**	Index 11: Start number for mode 4 of the voltage values set with index 8 Range: 1...100
[SOURce:]FUNCtion:GENerator:MPP:INDex␣12 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR1>**	Index 12: End number for mode 4 of the voltage values set with index 8 Range: 1...100
[SOURce:]FUNCtion:GENerator:MPP:INDex␣13 **[SOURce:]FUNCtion:GENerator:MPP:DATa[?]␣<NR1>**	Index 13: Number of repetitions for mode 4 Range: 0...65535

5.4.13.2 Control of the MPP tracking

The MPP tracking is started or stopped with a dedicated command. Independent from the DC input/terminal condition of the device or channel before the start, it would automatically switch the DC input/terminal on when sending the RUN command.

Command	Description
[SOURce:]FUNCtion:GENerator:MPP:STATe[?]␣{RUN | STOP} **[SOURce:]FUNCtion:GENerator:MPP:STATe?**	**RUN** = Runs the MPP tracking in the configured mode **STOP** = Stops MPP tracking anytime, no matter if there is a positive result or not **?** = Read the tracking status

The tracking status, as it can be read with command **FUNC:GEN:MPP:STAT?**, returns the current status as **RUN** (running) or **STOP** (stopped). When not intentionally stopped during the run, the meaning of status **STOP** slightly differs, depending on the chosen mode:

Mode 1: **STOP** means, the MPP has been found (positive result) and the tracking has been finished

Modes 2 and 3: These modes don't stop automatically, so **STOP** only returns the status as set by **:STAT** command

Mode 4: **STOP** means, the user curve has been processed the defined number of cycles

5.4.13.3 Reading results

Some of the MPP tracking modes deliver readable test results, which can be read after the test has stopped, also using **:INDex** and **:DATa** commands.

Command	Description
[SOURce:]FUNCtion:GENerator:MPP:INDex␣7 **[SOURce:]FUNCtion:GENerator:MPP:DATa?**	Index 7: Resulting MPP Reads three values (Uact, Iact, Pact) which define the found MPP (modes MPP1, MPP2 and MPP4)
[SOURce:]FUNCtion:GENerator:MPP:INDex␣9 **[SOURce:]FUNCtion:GENerator:MPP:LEVel[?]␣<NR1>** **[SOURce:]FUNCtion:GENerator:MPP:DATa?**	Index 9: Measured results of mode MPP4 Selects measured data to read: 1...100 Reads the data (three values: Uact, Iact, Pact)

5.4.14 Commands for alarm management

In remote control operation it's also important to manage alarms correctly, specifically device alarms such as OVP or OT. This can be done the same way as in manual control. When using SCPI command language, device alarms are typically indicated in a status register which can be polled.

5.4.14.1 Reading device alarms

Reading device alarms should be programmed in user-defined intervals, by querying the Questionable status registers by their sub registers **:CONDITION** or **:EVENT**. The commands **STAT:QUES:COND?** or **STAT:QUES?** or **STAT:QUES:EVEN?** would return a value that represents certain bits (see *"5.4.2. Status registers"*), indicating various statuses. When a bit is set, it means a certain alarm is present. Refer to the device's user manual for details about device alarms.

5.4.14.2 Acknowledging device alarms

In order to make the user take notice of device alarms, they require to be acknowledged after the cause has been removed. Acknowledgment will delete alarms from the status register and should only be done after they have been recorded. To delete/acknowledge an alarm, the command **SYST:ERR?** or **SYST:ERR:ALL?** is used, which also serves to query communication errors. In case one or multiple alarms are still present, they won't be cleared from the register. In this case the bit in CONDITION wouldn't change, but the one in EVENT.

5.4.14.3 Alarm counters

These counters count alarm occurrences since the moment the device was powered. They can be read by command anytime and are not stored when the device is switched off and are also not purged by reading.

Power supplies and electronic loads support these commands:

Command	Description
SYSTem:ALARm:COUNt:OVOLtage?	Counts overvoltage alarms (OVP, adjustable threshold)
SYSTem:ALARm:COUNt:OTEMperature?	Counts overtemperature alarms (OT, not adjustable)
SYSTem:ALARm:COUNt:OPOWer?	Counts overpower alarms (OPP, adjustable threshold)
SYSTem:ALARm:COUNt:OCURrent?	Counts overcurrent alarms (OCP, adjustable threshold)
SYSTem:ALARm:COUNt:PFAil?	Counts power fail alarms (PF, not adjustable)
SYSTem:ALARm:COUNt:SHARebusfail?	Count Share bus fail alarms (SF, not adjustable)

Bidirectional power supplies support these commands additionally:

Command	Description
SYSTem:SINK:ALARm:COUNt:OPOWer?	Counts overpower alarms (OPP, adjustable threshold), if they occur during sink operation, for distinction from source operation
SYSTem:SINK:ALARm:COUNt:OCURrent?	Counts overcurrent alarms (OCP, adjustable threshold), if they occur during sink operation, for distinction from source operation

5.4.14.4 Example

You are running the device in remote control and poll the alarm status with **STAT:QUES:COND?** (single channel device) or **STAT:QUES:COND?␣(@2)** (channel 2 of a multi-channel device) command in a certain interval and you always receive value 3072. This is the sum of the bit values of bits 10 (remote control on) and 11 (DC terminal/input on). It tells you that remote control is active and the DC terminal/input is switched on. Let's say, later a device alarm occurs caused by the unit overheating. When reading the questionable register the next time, bit 3 should indicate the OT alarm for you to take notice. Additionally, the DC terminal/input would be signaled as "switched off" by the returned value being 1032.

5.4.15 Commands for the extended PV simulation

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	–	✓

The extended Photovoltaics simulation according to **DIN EN 50530** is a function based on the XY generator, but different to the simple PV simulation (see *5.4.12.6*), because here the user doesn't have to deal with table data and XY curve loading. Since PV simulation is that of a source, the function is generally only available with power supplies and is part of the feature "Function generator". The focus of this function is on the higher configurability which is achieved by more user-accessible parameters from which the device would calculate a PV curve either once when the simulation starts or repeatedly when the so-called day trend is run, where curve updates are required.

All simulation related parameters, as configurable with the below listed commands, are specified and described in the norm paper. The norm paper is considered as additional reference to further information.

5.4.15.1 General configuration

Command	Description
[SOURce:]FUNCtion:PHOTovoltaics:MODe˽{OFF | ET | UI | DAYET | DAYUI} **[SOURce:]FUNCtion:PHOTovoltaics:MODe?**	General mode selection for the PV simulation **OFF** = Simulation mode off (default) **ET** = Continuous mode, temperature and irradiation can be varied during simulation **UI** = Continuous mode, voltage and current of the MPP can be varied during simulation **DAYET** = Day trend mode, no values can be varied, but it runs on a certain number of data sets that are loaded into the device or into a specific channel. A data set consists of an index, a temperature value, an irradiation value and a dwell time **DAYUI** = Same as DAYET, but here the data set consists of an index, MPP voltage, MPP current and a dwell time
[SOURce:]FUNCtion:PHOTovoltaics:IMODe˽{MPP | ULIK} **[SOURce:]FUNCtion:PHOTovoltaics:IMODe?**	Input mode for the basic parameters of the simulated panel (applies to all modes selectable with :**MODe** command, also see matrix and examples in *5.5.3*) **MPP** = The base values to calculate the PV curve from are entered as Umpp and Impp. These values are adjustable simulation mode **UI** (default) **ULIK** = The base values to calculate the PV curve from are entered as U_{OC} (open circuit voltage) and I_{SC} (short-circuit current). These values are adjustable in simulation mode **ET**
[SOURce:]FUNCtion:PHOTovoltaics:TECHnology˽{MAN | CSI | THIN} **[SOURce:]FUNCtion:PHOTovoltaics:TECHnology?**	Panel technology preselection. Determines whether some simulation parameters are fixed or accessible **MAN** = Manual mode (all parameters unlocked) **CSI** = cSi technology panel (default) **THIN** = thin film technology panel

5.4.15.2 Day trend mode configuration

The below listed commands can only be used if one of the day trend modes **DAYET** or **DAYUI** (see above) has been set before or using them would return an error.

It's recommended to clear the old day data with :DAY CLEAR command before loading a new set, especially if the new set is shorter.

Commands	Description
[SOURce:]FUNCtion:PHOTovoltaics:DAY:INTerpolate˽{ON | OFF} **[SOURce:]FUNCtion:PHOTovoltaics:DAY:INTerpolate?**	**ON** = Interpolation on **OFF** = Interpolation off (default)
[SOURce:]FUNCtion:PHOTovoltaics:DAY:MODe˽{READ | WRITe} **[SOURce:]FUNCtion:PHOTovoltaics:DAY:MODe?**	Day trend mode data access type **READ** = Read only (default) **WRITe** = Write only
[SOURce:]FUNCtion:PHOTovoltaics:DAY˽CLEar	Clears all previously loaded day trend data sets

Commands	Description
[SOURce:]FUNCtion:PHOTovoltaics:DAY:INDex␣<NR1> **[SOURce:]FUNCtion:PHOTovoltaics:DAY:INDex?**	Selects the data index (1...100,000) before re-reading of day trend data sets. When writing day trend data sets, this index values is ignored. For that use the index number at **FUNC:-PHOT:DAY:DAT** command instead.
[SOURce:]FUNCtion:PHOTovoltaics:DAY:DAT␣<NR1>, <NR2>, <NR2>, <NR1> **[SOURce:]FUNCtion:PHOTovoltaics:DAY:DAT?**	Writes a day trend data sets (4 values) or reads it back from the prior selected index. Depending on the chosen day trend mode, different data must be provided when writing.
	Mode **DAYET**:
	1. value = index, range: 1- 100,000
	2. value = irradiation in W/m², range: 0-1500
	3. value = temperature in °C, range: -40...+80
	4. value = Dwell time of the index in milliseconds, range: 500...1,800,000 (^=0.5s...0.5h)
	Mode **DAYUI**:
	1. value = index, range: 1-100,000
	2. value = Umpp in V, range: 0...rated voltage
	3. value = Impp in A, range: 0...rated current
	4. value = Dwell time of the index in milliseconds, range: 500...1,800,000 (^=0.5s...0.5h)

5.4.15.3 Data recording

The device can record data while the PV simulation is running in any mode. It records up to 576,000 data sets with 6 values each (actual values of U, I, P and MPP values U, I, P). The recording can be started with the simulation or while it runs. Once the internal memory is filled, it overwrites from the beginning and the number of recorded data sets (:REC:NUM?) is reset to 0 . There is one new data set recorded every 100 ms, so that it covers a total time of exactly 16 hours.

The recording is either stopped at the end of the simulation or on purpose by the user. After the stop, the recorded data can be read data set by data set. In case they are needed to be saved, they should be read as long as the unit is powered, because the recorded data isn't stored.

Command	Description
[SOURce:]FUNCtion:PHOTovoltaics:RECord:ACTive␣{ENABle | DISable} **[SOURce:]FUNCtion:PHOTovoltaics:RECord:ACTive?**	Data recording **ENABle** = activated **DISable** = deactivated (default)
[SOURce:]FUNCtion:PHOTovoltaics:RECord␣CLEar	Clear recorded data
[SOURce:]FUNCtion:PHOTovoltaics:RECord:NUMBer?	Returns number of already recorded data sets (0...576,000)
[SOURce:]FUNCtion:PHOTovoltaics:RECord:INDex␣<NR1> **[SOURce:]FUNCtion:PHOTovoltaics:RECord:INDex?**	Set or read the index number (1...576,000) prior to read a data set with **:DATa?** command
[SOURce:]FUNCtion:PHOTovoltaics:RECord:DATa?	Read data set X from the previously selected index. The device will then return following values separated by commas, representing a snapshot from a certain time: 1. value = Index number 2. value = Actual voltage on the DC output 3. value = Actual current on the DC output 4. value = Actual power on the DC output 5. value = Umpp (voltage in the MPP) 6. value = Impp (current in the MPP) 7. value = Pmpp (power in the MPP)

After selecting the index with FUNC:PHOT:REC:IND, prior to reading the data set, it requires some time (<5 ms) to pass before the device can return the data set of the index from an internal buffer. Being too early with the request command will cause the device to write an error into the error queue. After data reception, the correct data can be verified by comparing the index number in the data set with the index you selected with FUNC:PHOT:REC:IND.

After the simulation start, the device will calculate the first PV curve. This takes about 500 ms. However, the first data set is already recorded 100 ms after the simulation start, so the first 3-4 data sets will be erroneous. This wouldn't be the case if the recording is started at least 500 ms after the simulation start.

5.4.15.4 Statuses from the simulation

Status commands and those which read result values from the simulation can be used at any time, but it's recommend to carefully chose the moment and order of use.

Command	Description
[SOURce:]FUNCtion:PHOTovoltaics:MPP:VOLTage?	Voltage in the MPP, in Volt. The MPP results from the PV simulation curve, which is calculated by the given simulation settings. The voltage can be between 0 and the rated device voltage.
[SOURce:]FUNCtion:PHOTovoltaics:MPP:CURRent?	Current in the MPP, in Ampere. Can be between 0 and rated current.
[SOURce:]FUNCtion:PHOTovoltaics:MPP:POWer?	Power in the MPP, in Ampere. Can be between 0 and rated power.
[SOURce:]FUNCtion:PHOTovoltaics:STATe?	PV simulation status **STOP** = Simulation has stopped normally, either due to user action or end of day trend **RUN** = Simulation running **ERROR MODE** = Simulation didn't start due to an error during PV curve calculation in simulation modes **ET** or **UI** **ERROR DAY** = Simulation didn't start due to an error during PV curve calculation in simulation modes **DAYET** or **DAYUI** **ERROR ALARM** = Simulation has stopped due to a device alarm **ERROR INTERPOLATION** = Simulation didn't start due to a wrong dwell time value in day trend data index 1
[SOURce:]FUNCtion:PHOTovoltaics:DAY:NUMBer?	Number of accepted day trend indexes. When transferring day trend data to the device, this counter counts up every time an index is successfully transferred and accepted. Can ve used to verify written data.
[SOURce:]FUNCtion:PHOTovoltaics:OCVoltage?	Open circuit voltage of the simulated solar panel, calculated with a formula according to the standard. The values is affected by the selected simulation mode, the standard panel parameters (see below) and the calculation factors from the chosen panel technology (also see below).
[SOURce:]FUNCtion:PHOTovoltaics:SCCurrent?	Short-circuit current of the simulated solar panel, calculated with a formula according to the standard. The values is affected by the selected simulation mode, the standard panel parameters (see below) and the calculation factors from the chosen panel technology (also see below).

5.4.15.5 Parameter commands

The commands listed below are used to set or read all the values required for the different PV simulation modes and the PV curve calculation. Not all commands can be written anytime. When trying to write a value it's important what simulation mode (ET, UI, DAYET, DAYUI) and what input mode (MPP, ULIK) are currently set. The matrix below indicates which commands are supported in what mode. A third setting, the technology (MAN, CSI, THIN) also determines if a specific parameter is locked from writing, but instead is internally setup with a value according to the DIN EN 50530 standard. Reading the parameters is possible anytime and in every mode.

Command	Writable in mode:				
	MPP	ULIK	MAN	CSI	THIN
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:FFU␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:FFU?** Fill factor for voltage (FF_U). Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	–	–
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:FFI␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:FFI?** Fill factor for current (FF_I). Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	–	–

Command	Writable in mode:				
	MPP	ULIK	MAN	CSI	THIN
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:ALPHa␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:ALPHa?** Temperature coefficient α (in 1/°C) for the short-circuit current. Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	—	—
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:BETA␣<NRf> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:BETA?** Temperature coefficient β (in 1/°C) for the open circuit voltage. Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: -1...<0	✓	✓	✓	—	—
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CU␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CU?** Scaling factor C_U for the open circuit voltage. Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	—	—
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CR␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CR?** Scaling factor C_R in m²/W or the open circuit voltage. Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	—	—
[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CG␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:FACTor:CG?** Scaling factor C_G in W/m² for the open circuit voltage. Only writable if the technology is set to MAN. Has impact on the PV curve calculation with the formula according to the norm paper. Range: >0...1	✓	✓	✓	—	—
[SOURce:]FUNCtion:PHOTovoltaics:STANdard:OCVoltage␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:STANdard:OCVoltage?** U_{OC} (open circuit voltage) of the simulated solar panel in V. Range: 0...rated voltage	—	✓	✓	✓	✓
[SOURce:]FUNCtion:PHOTovoltaics:STANdard:SCCurrent␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:STANdard:SCCurrent?** I_{SC} (short-circuit current) of the simulated solar panel in A. Range: 0...rated current	—	✓	✓	✓	✓
[SOURce:]FUNCtion:PHOTovoltaics:STANdard:MPP:VOLTage␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:STANdard:MPP:VOLTage?** Voltage in the MPP of the simulated solar panel, in V. Range: 0...rated voltage	✓	—	✓	✓	✓
[SOURce:]FUNCtion:PHOTovoltaics:STANdard:MPP:CURRent␣<NR2> **[SOURce:]FUNCtion:PHOTovoltaics:STANdard:MPP:CURRent?** Current in the MPP of the simulated solar panel, in A. Range: 0...rated current	✓	—	✓	✓	✓

5.4.15.6 Control commands

These commands are used to control the PV simulation, usually after a successful configuration. In some modes, one or two parameters are adjustable while the simulation is running. Any change of parameter requires re-calculation of the PV curve which overwrites previous curve data. Depending on what MPP on the curve the simulation currently is, the MPP will shift after a certain calculation and response time.

Command	Description
[SOURce:]FUNCtion:PHOTovoltaics:STATe␣{RUN | STOP} **[SOURce:]FUNCtion:PHOTovoltaics:STATe?**	Start/stop simulation **RUN** = Triggers PV curve calculation and succeeding simulation start, if there is no error occurring. If data recording is activated, it will also start **STOP** = Simulation and possibly running data recording are stopped
[SOURce:]FUNCtion:PHOTovoltaics:TEMPerature␣<NRf> **[SOURce:]FUNCtion:PHOTovoltaics:TEMPerature?**	Only available in mode **ET**: Solar module temperature in °C. Range: -40...+80
[SOURce:]FUNCtion:PHOTovoltaics:IRRadiation␣<NR1> **[SOURce:]FUNCtion:PHOTovoltaics:IRRadiation?**	Only available in mode **ET**: Irradiation in W/m². Range: 0-1500

5.4.15.7 Error situations

An error situation can occur when the configured simulation can't be started or it has started already and was running for a while, but then unexpectedly stopped. Command **FUNCtion:PHOTovoltaics:STATe?** (see section *5.4.15.4*) can help in both cases to identify the cause of the error.

Following generally applies:

- Once stopped for any reason, the simulation can't be continued
- Data from the data recording feature can be read during the simulation or after the stop, as long as the device remains powered
- All configuration parameters are not stored and are reset when power-cycling the device

5.4.16 Commands for the battery test function

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	–

Electronic load and also bidirectional power supply series feature a battery test function. The remote control of that feature and the handling is almost the same as with manual control on the device's HMI. Additionally to the separate battery test modes for charging (not with electronic loads) and discharging, bidirectional devices series offer a combined mode of both called the "Dynamic test". This isn't to be confused with the separate "Dynamic discharge" test as named on the HMI.

5.4.16.1 Configuration commands

The battery test configuration can be done in the same sequence of commands as listed by the tables below. The very first thing to do is to always select the test mode. Details about the battery test modes can be found in the user manual of the device.

Command	Description
[SOURce:]BATTery:MODe␣{IDLE | STATic | CHARge | DYNamic | COMBined} **[SOURce:]BATTery:MODe?**	Selects battery test mode or queries the selected mode: **IDLE** = no mode selected, also used to leave battery test mode **STATic** = Select mode "Static discharge" **DYNamic** = Select mode "Dynamic discharge" Additional battery test modes for bidirectional devices: **CHARge** = Select mode "Static charge" **COMBined** = Select "Dynamic test" mode which combines static charge and static discharge modes

The following commands only work for sending or reading values after any of the discharge modes **STATIC** or **DYNAMIC** has been selected:

Command	Description
[SOURce:]BATTery:CURRent␣<NR2> **[SOURce:]BATTery:CURRent?**	Only for static discharge mode: Set the (discharge) current in Amperes. The command doesn't work in mode **DYNAMIC**. Range: 0...I-max
[SOURce:]BATTery:POWer␣<NR3> **[SOURce:]BATTery:POWer?**	For static and dynamic discharge mode: Set the maximum power in Watts. Constant power regulation can override constant current, so it may affect the actual discharge current according to I = P/U to be lower than defined by **BATT:CURR**. Range: 0...P-max

[SOURce:]BATTery:RESistance␣<NR2>
[SOURce:]BATTery:RESistance?
	For static or dynamic discharge mode:
	Switches resistance mode for the test on or off and sets the resistance value in Ohms. Constant resistance regulation can override constant current, so it may affect the actual discharge current according to I = U/R to be lower than defined by **BATT:CURR**.
	NR2 = 0 = resistance mode off
	NR2 = min. R ... max. R = set resistance

The commands below are for both discharge test modes and define one or several stop conditions which can cause the battery test to stop automatically if any of these conditions is triggered:

Command	Description
[SOURce:]BATTery:DISCharge:VOLTage␣<NR2> **[SOURce:]BATTery:DISCharge:VOLTage?**	Defines the discharge voltage in Volt. Once the battery voltage reaches this threshold, the test will stop. Range: 0...U-max
[SOURce:]BATTery:DISCharge:CAPacity␣<NR2> **[SOURce:]BATTery:DISCharge:CAPacity?**	Defines the max. battery capacity to consume before the test can either stop or cause the device to write a message into its display and into the battery test status. Range: 0...99999.99 Ah

Command	Description
[SOURce:]BATTery:ACTion:CAPacity␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:CAPacity?**	Defines the action upon reaching the limit as defined by **BATT:DIS:CAP**. **NONE** = no action (the limit will be ignored) **SIGNal** = test continues, but the device will signal that the limit has been reached **END** = test will be stopped
[SOURce:]BATTery:DISCharge:TIME␣<Time2> **[SOURce:]BATTery:DISCharge:TIME?**	Defines the max. time to run the battery test, after which the test can either stop or cause the device to write a message into its display and the status. Range: 00:00:00 ... 10:00:00
[SOURce:]BATTery:ACTion:TIME␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:TIME?**	Same as **BATT:ACT:CAP**, but here for the max. test time as it can be defined with **BATT:DIS:TIME**.

When using the dynamic discharge test mode, a pulsed current is generated which is defined by a rectangle with adjustable amplitude and duty cycle. Following commands only work in mode **DYNAMIC**:

Command	Description
[SOURce:]BATTery:INDex␣<NR1> **[SOURce:]BATTery:INDex?**	Select one out of 4 parameters (range: 0...3) from which the rectangle for the pulsed current is defined: **0** = Level 1 of the amplitude Range: 0...I-max (adjustment limit) **1** = Level 2 of the amplitude Range: 0...I-max (adjustment limit) **2** = Time of level 1 Range: 0...36000 s **3** = Time of level 2 Range: 0...36000 s
[SOURce:]BATTery:PULSe␣<NR2> **[SOURce:]BATTery:PULSe␣<Time3>** **[SOURce:]BATTery:PULSe?**	Sets or reads the value from the index which was previously selected with **BATTery:INDex**. Values for the amplitude of current can be given in format <NR2> while the time values for indexes 2 and 3 require format <Time3>.

Bidirectional series, which can also charge a battery in source mode, feature another test mode: static charge. This mode works the same way as the static discharge, but in inverse direction.

Command	Description
[SOURce:]BATTery:CHARge:VOLTage␣<NR2> **[SOURce:]BATTery:CHARge:VOLTage?**	Defines the charging voltage in Volts. The level depends on whether you want to have a precharge, boost charge or trickle charge. Range: 0...U-max
[SOURce:]BATTery:CHARge:CURRent␣<NR2> **[SOURce:]BATTery:CHARge:CURRent?**	Set the charge current in Amperes. The actual charge current will sink over time. Range: 0...I-max
[SOURce:]BATTery:CHARge:STOP:CURRent␣<NR2> **[SOURce:]BATTery:CHARge:STOP:CURRent?**	Set the charging end current in Amperes. Once this threshold is reached, the test will stop. Range: 0...I-max
[SOURce:]BATTery:CHARge:STOP:CAPacity␣<NR2> **[SOURce:]BATTery:CHARge:STOP:CAPacity?**	Defines the max. battery capacity to charge before the test can either stop or cause the device to write a message into its display and the status. Range: 0...99999.99 Ah
[SOURce:]BATTery:CHARge:STOP:TIME␣<Time2> **[SOURce:]BATTery:CHARge:STOP:TIME?**	Defines the max. time to run the battery test, after which it can either stop or cause the device to write a message into its display and the status. Range: 00:00:00 ... 10:00:00

Command	Description
[SOURce:]BATTery:ACTion:CAPacity␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:CAPacity?**	Defines the action upon reaching the limit as defined by **BATT:CHAR:STOP:CAP**. **NONE** = no action (the limit will be ignored) **SIGNal** = test continues, but the device will signal that the limit has been reached **END** = test will be stopped
[SOURce:]BATTery:ACTion:TIME␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:TIME?**	Same as **BATT:ACT:CAP**, but here for the max. test time as it can be defined with **BATT:CHAR:STOP:TIME**.

The so-called **dynamic test**, which combines charging and discharging of a battery, is only available with bidirectional device series, as they can switch back and forth between source and sink mode. Details about this mode can be found in the user manual of a bidirectional device.

Since the test combines the two single modes static charge and static discharge, some of their commands are re-used here for configuration of the single test phases. To have a better overview about the parameters which have to be configured for the test, some commands are copied in the tables below.

Commands for the charging phase configuration, given that **BATTery:MODe␣COMBined** has been set:

Command	Description
[SOURce:]BATTery:CHARge:VOLTage␣<NR2> **[SOURce:]BATTery:CHARge:VOLTage?**	Defines the charging voltage in Volts. The level depends on whether you want to have a precharge, boost charge or trickle charge. Range: 0...U-max
[SOURce:]BATTery:CHARge:CURRent␣<NR2> **[SOURce:]BATTery:CHARge:CURRent?**	Set the charge current in Amperes. The actual charge current will sink over time. Range: 0...I-max
[SOURce:]BATTery:CHARge:STOP:CURRent␣<NR2> **[SOURce:]BATTery:CHARge:CURRent?**	Set the charging end current in Amperes. Once this threshold is reached, the phase will stop. Range: 0...I-max
[SOURce:]BATTery:CHARge:TIME␣<Time2> **[SOURce:]BATTery:CHARge:TIME?**	Defines the max. time (in seconds) to run the phase. When reaching the defined time (max. 10 h), the phase will be ended. Alternatively, the charging end current can end the phase. Range: 00:00:01...10:00:00

Commands for discharging phase configuration, given that **BATTery:MODe␣COMBined** has been set:

Command	Description
[SOURce:]BATTery:COMBined:CURRent␣<NR2> **[SOURce:]BATTery:COMBined:CURRent?**	Set the discharge current in Amperes. Range: 0...I-max
[SOURce:]BATTery:DISCharge:VOLTage␣<NR2> **[SOURce:]BATTery:DISCharge:VOLTage?**	Set the charging end voltage (U_{DV}) in Volt. Once this threshold is reached, the phase will stop. Range: 0...U-max
[SOURce:]BATTery:DISCharge:TIME␣<Time2> **[SOURce:]BATTery:DISCharge:TIME?**	Defines the max. time (in seconds) to run the phase. When reaching the defined time (max. 10 h), the phase will be ended. Alternatively, the charging end current can end the phase. Range: 00:00:01...10:00:00

Commands for the remaining configuration, given that **BATTery:MODe␣COMBined** has been set:

Command	Description
[SOURce:]BATTery:COMBined:STOP:CAPacity␣<NR2> **[SOURce:]BATTery:COMBined:STOP:CAPacity?**	Defines the max. battery capacity to charge and/or discharge before the test can either stop or cause the device to write a message into its display and the status. This is a global value for both phases. Range: 0...99999.99 Ah
[SOURce:]BATTery:COMBined:STOP:TIME␣<Time2> **[SOURce:]BATTery:COMBined:STOP:TIME?**	Defines the max. time to run the test. When reaching the defined time, the test will be ended. This is a global value for both phases. Range: 00:00:01 ... 10:00:00

Command	Description
[SOURce:]BATTery:ACTion:CAPacity␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:CAPacity?**	Defines the action upon reaching the limit as defined by **BATT:COMB:STOP:CAP**. **NONE** = no action (the limit will be ignored) **SIGNal** = test continues, but the device will signal that the limit has been reached **END** = test will be stopped
[SOURce:]BATTery:ACTion:TIME␣{ NONE | SIGNal | END} **[SOURce:]BATTery:ACTion:TIME?**	Same as **BATT:ACT:CAP**, but here for the max. test time as it can be defined with **BATT:COMB:STOP:TIME**.
[SOURce:]BATTery:COMBined:TIME␣<Time3> **[SOURce:]BATTery:COMBined:TIME?**	Rest time between phases (in seconds). Range: 1...36000
[SOURce:]BATTery:COMBined:STARt␣{CHARge | DIS-Charge} **[SOURce:]BATTery:COMBined:STARt?**	Defines with which phase the test will start, either with the charging phase (**CHARge**) or discharging phase (**DIS-Charge**). After a phase has ended, the other one will run through. It means that both phases are always run through at least once per test.
[SOURce:]BATTery:COMBined:CYCLes␣<NR1> **[SOURce:]BATTery:COMBined:CYCLes?**	Defines how many times the test is run through. The test itself will only stop when either reaching the defined number of cycles or when any of the other stop criteria becomes true. Range: 0 (infinite cycles) or 1...999

5.4.16.2 Control and status commands

After successful battery test configuration the test can be started. Unless there will be no unexpected event such as a device alarm, the battery test will run and later stop when at least one of the defined stop conditions has been reached.

Command	Description
[SOURce:]BATTery:STATe␣{ RUN | STOP } **[SOURce:]BATTery:STATe?**	Starts the test or stops it manually and anytime before reaching a stop condition. **RUN** = Start test **STOP** = Stop test immediately
[SOURce:]BATTery:CONDition?	Queries the test condition. This can be done anytime, i.e. before, during or after the test. **IDLE** = test has not yet been started or has been stopped manually or due to an alarm **RUN** = test is running **FINISHED** = Test is finished as expected **ERROR** = Test stopped due to device alarm Further conditions: **SIGNAL AH** = Max. allowed number of Ah reached, test continues **SIGNAL TIME** = Max. allowed test time reached, test continues **END AH** = Max. allowed number of Ah reached, test stopped **END TIME** = Max. allowed test time reached, test stopped Further conditions (only available with bidirectional devices): **RUN, CHARGING** = test is running in the charging phase **RUN, DISCHARGING** = test is running in the discharging phase **RUN, RESTING** = test is resting between the two test phases

5.4.16.3 Test result commands

After the test end, a few values can be read which represent the test result.

Command	Description
[SOURce:]BATTery:TEST?	Queries the test results. It should return a string with three values. Can also be queried while the test is running: 1. item: Consumed and/or supplied capacity in Ah 2. item: Consumed and/or supplied energy in Wh 3. item: Elapsed battery test time in format <Time2>

Command	Description
[SOURce:]BATTery:TEST {RESet | RESETAH | RESETWH | RESETTIME}	Used to reset all or specific test result values. This can be used to reset result data in between tests. The same effect is achieved when leaving battery test mode and entering it again. **RESet** = reset all result data to zero **RESETAH** = only resets the Ah value to zero **RESETWH** = only resets the Wh value to zero **RESETTIME** = only resets the time counter to zero

5.4.16.4 Tips and hints

- Consumed capacity isn't deducted from supplied capacity, the same applies to energy.
- In order to get intermediate Ah and Wh values from a single phase or cycle of a longterm dynamic test, these two values could be read before and after a phase, perhaps during the rest period, to calculate the difference.
- If the max. 10 hours of a phase in dynamic test don't suffice, the whole dynamic test could also be achieved by moving the flow control into a custom control software which would then probably only use and configure the "Static Charge" and "Static Discharge" test modes, combining them with rest periods. Those single tests allow for a theoretically infinite testing time.
- The so called "dynamic test", as only available with bidirectional devices, combines one or several charging and discharging phases into a test flow. The counted values of Ah and Wh are always only counted up during the phase, so it's not possible to retrieve at the end the actual capacity or energy supplied to the battery.

5.5 Example applications

5.5.1 Configure and control master-slave / master-auxiliary with SCPI

The feature called master-slave (short: MS) with 10000 series and master-auxiliary (short: MA) with 20000 series supports full remote configuration and control. In an MS/MA system, only the master unit is remotely controlled, while the slave/auxiliary devices are not connected to a PC or other kind of control unit, except for the moment when they're configured remotely as slave/auxiliary. It's therefore recommended to configure the MS/MA system on the control panels of the units and only put the master into remote control via any software. Even if you would configure all units manually on the control panel, the remote control software could later read the status of the MS/MA initialization from the master. The initialization of the MS/MA system is done automatically every time the master is powered, but can be triggered and repeated by command.

The below shown example is also applicable for 20000 series device, but it's necessary to replace :MS with :MA in the master-auxiliary commands. Also see 5.4.9.

Let's assume the following example configuration: five power supplies **PSI 10080-510 3U** (80 V, 510 A, 15 kW) in parallel. The master has to display itself as an 80V, 2550 A , 75 kW and 1 Ω (max. resistance) unit after successful configuration and initialization. These values are also the temporary new ratings of the MS system. The same way as with manual control, the "Limits" and set values can be adjusted in 0...102% of the rated value, while protection values have a range of 0..110% (most series) or 0...103% (specific load series).

The exemplary step-by-step guide below is separated into steps, because some steps are optional.

Part 1a: Configure the master

1. Activate remote control: **SYST:LOCK␣ON**
2. Activate master-slave mode: **SYST:MS:ENABLE␣ON** (20000 series: **SYST:MA:ENABLE␣ON**)
3. Define the unit as master: **SYST:MS:LINK␣MASTER** (20000 series: **SYST:MA:LINK␣MASTER**)

Part 1b: Configure the slave(s), in case it's connected to the controlling unit (PC, PLC etc.)

4. Activate remote control: **SYST:LOCK␣ON**
5. Activate master-slave mode: **SYST:MS:ENABLE␣ON** (20000 series: **SYST:MA:ENABLE␣ON**)
6. Define the unit as slave: **SYST:MS:LINK␣SLAVE** (20000 series: **SYST:MA:LINK␣AUX**)

If there is more than one slave, repeat steps 5-7 for the other slave(s)/aux units.

Part 2: Initialize the MS system

7. Activate remote control if steps 1-6 were not processed, because the system was already configured: **SYST:LOCK␣ON**
8. Trigger initialization, then wait a few seconds: **SYST:MS:INIT** (20000 series: **SYST:MA:INIT**)

Part 3: Further, optional steps

9. Query the initialization status from the master, in order to analyze it: **SYST:MS:COND?** (20000 series: **SYST:MA:COND?**)
10. Query the number of units initialized for the MS/MA system: **SYST:MS:UNIT?** (20000 series: **SYST:MA:UNIT?**)
11. Query the nominal current of the MS/MA system: **SYST:NOM:CURR?**
12. Query the nominal power of the MS/MA system: **SYST:NOM:POW?**
13. Query the maximum resistance of the MS/MA system: **SYST:NOM:RES:MAX?**
14. Query the minimum resistance of the MS/MA system: **SYST:NOM:RES:MIN?**
15. Configure protection values, for example OCP: **CURR:PROT␣400**
16. Configure events, for example,
    - set OCD to 2100 A: **SYST:CONF:OCD␣2100**
    - then define the alarm type for OCD to "warning": **SYST:CONF:OCD:ACT␣WARNING**

The adjustment limits ("Limits") require extra treatment, because they are tied to the set values. Means, with the set values being reset to defaults during the MS/MA initialization, for example the set value of current would be at maximum and thus the related adjustment limit I_{Max} can't be set lower than this without prior changing the set value.

17. Narrow the adjustable range of values, for example limit the max. current set value to 2200 A
    - First, set the current value down to anything lower than the desired limit, like the minimum: **CURR␣MIN**
    - Second, set the adjustment limit to the value translated for the master unit: **CURR:LIM:HIGH␣2200**

With these settings applied, the current should be at 0, because the lower adjust limit has not yet been changed. The current will be monitored for the threshold of 2100 A by the event system and since it's adjustable up to 2200 A, the true current might exceed the threshold and cause an OCD event, which would only generate a warning on screen, but not switch off the DC terminal.

18. To start working with your MS system, switch the DC output on: **OUTP˽ON**

The system will remain configured and keep the settings when power-cycling it. The master unit has to initialize the MS/MA and the slaves/aux units at least one time after power-up. The status of the first automatic initialization can be read from the master in your custom software and depending on the result, the software could trigger further steps like the ones listed above, probably from at least step 8 or if required even from step 1.

5.5.2 Programming examples for the function generator

5.5.2.1 General command sequence for the arbitrary generator

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

Let’s say you wanted to apply a sine wave with 30 A amplitude and 10 Hz frequency for 60 s to the input current of an electronic load or the sink mode of a bidirectional device. This can be achieved by setting up just one sequence point, for example sequence point number 12. Because this is about DC sine wave, the amplitude also requires an offset. The amplitude is usually understood as the difference between the base line and the peak point of the sine wave. The base line is here defined by Start(DC) and End(DC) values of a sequence point. Also see the device's user manual about the definition of these parameters, as well as section *5.4.12.3*. In this example, the offset, which defines the base line, then has to be at least 30 A, so we perhaps take 50 A. This will result in a DC input current varying sinusoidally between 20 A and 80 A.

The sine wave, when applied to DC voltage or current, emulates AC characteristics and thus requires to set at least indexes 0, 1, 2, 3, 5, 6 and 7, according to the table in *5.4.12.3*. As long as no specific start angle is required, index 4 can be skipped, because the default value is 0°.

Setting the global set values to maximum or any other value is also necessary when using the function generator, especially when running multiple devices in master-slave/master-auxiliary where those set values are used to limit the working range of the slave/aux units while they're controlled via the Share-Bus.

When loading the arbitrary function generator with sequence point data, the following rules apply:

- When also using the AC part of a sequence point, i. e. when generating a sine wave, the value(s) for the DC part must be set first
- When using ramps, i. e. DC start value is different to DC end value, there is a certain minimum ratio between the voltage difference and the time per sequence point, here called min. slew rate; this applies to many series, but not anymore to 10 k series devices which already have a specific minimum KE firmware version installed -> should you receive an error when writing sequence data it's most likely due to the min. slew rate not met
- The time of the sequence point should be written as last values because it's used to trigger the slew rate check

Given that the device is already in remote control and the DC input or DC output is off, the following command sequence would be necessary:

No.	Command	Description	Register
1	**FUNC:GEN:SEL CURR**	Selects arbitrary generator for current. By sending this command the device will switch to function generator mode	852
2	**FUNC:GEN:WAVE:LEV˽12**	Selects the 12th sequence point for writing values	1092
3	**FUNC:GEN:WAVE:IND˽7**	Select index 7: Sequence point time	
4	**FUNC:GEN:WAVE:DAT˽60**	Set sequence point time to 60 s	1092
5	**FUNC:GEN:WAVE:IND˽5**	Select index 5: Start value of DC part or AC offset	
6	**FUNC:GEN:WAVE:DAT˽50**	Set wave offset to 50 A	1092
7	**FUNC:GEN:WAVE:IND˽6**	Select index 6: End value of DC part or AC offset	
8	**FUNC:GEN:WAVE:DAT˽50**	Set wave offset to 50 A. If the offset shall not change during the function run, end and start value have to be identical.	1092
9	**FUNC:GEN:WAVE:IND˽2**	Select index 2: Start frequency of sine wave	
10	**FUNC:GEN:WAVE:DAT˽10**	Set start frequency to 10 Hz	1092
11	**FUNC:GEN:WAVE:IND˽3**	Select index 3: End frequency of sine wave	

No.	Command	Description	Register
12	**FUNC:GEN:WAVE:DAT␣10**	Set end frequency to 10 Hz. If the frequency shall not change during the function run, end and start value have to be identical.	1092
13	**FUNC:GEN:WAVE:IND␣0**	Select index 0: Start value of sine wave amplitude	
14	**FUNC:GEN:WAVE:DAT␣30**	Set amplitude to 30 A	1092
15	**FUNC:GEN:WAVE:IND␣1**	Select index 1: End value of sine wave amplitude	
16	**FUNC:GEN:WAVE:DAT␣30**	Set amplitude to 30 A. If the amplitude shall not change during the function run, end and start value have to be identical.	1092
17	**FUNC:GEN:WAVE:END␣12**	Set end sequence point to 12	860
18	**FUNC:GEN:WAVE:STAR␣12**	Set start sequence point to 12	859
19	**FUNC:GEN:WAVE:NUM␣1**	Set number of sequence point cycles to 1, because that one sequence point will already run for 60 s. Alternatively, it's possible to define 1 s for the sequence point time and let it run through 60 cycles.	861
20	**FUNC:GEN:WAVE:SUBM**	Load the parameters from above into the function generator	

Now you should also define the global set values of U, I and P so they don't interfere with the expected function run.

In this example with a current sink (electronic load), it's recommended to set the voltage to 0 V, the power to maximum and the current to 105% or higher of the peak that would result from the sine wave current. The current set value here is, however, only required if the target device is an MS/MA system, where the current value is transferred to the slave/aux units.

No.	Command	Description	Register
21	**VOLT␣0**	Set voltage to 0 V, so the device can clearly operate in current control mode	500
22	**optional: CURR␣MAX**	Set the current to maximum for MS/MA system and their slaves/aux units	501
23	**POW␣MAX**	Power to max, independent from the device model	502 (498)

The function generator is now configured and sequence 12 is set up. You can start the function run now and control it remotely:

No.	Command	Description	Register
24	**INP␣ON** or **OUTP␣ON**	Switch the DC input or DC terminal of your device on. INP ON for electronic loads, OUTP ON for all other device types	405
25	**FUNC:GEN:WAVE:STAT␣RUN**	Start the function with **RUN**. After 60 s, the function will stop.	850
26	**FUNC:GEN:WAVE:STAT␣STOP**	Optional: stop/abort function run anytime. The DC input/output will remain on at first and can be switched off with the dedicated command, if necessary	850
27	**FUNC:GEN:SEL␣NONE**	Optional: leave function generator mode with **NONE**	852

5.5.2.2 Command sequence for the XY generator

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

Configuration and loading of table data for the XY generator is very similar to the procedure of the arbitrary generator. Assumption: you want to have the DC input current of an electronic load react to the input voltage. This is were the XY generator is well suited with its IU function.

The IU table with its data determines the current to draw from the source for the entire input voltage range, which is resolved in 4096 values. With this you can define everything you want, like for example the current to remain 0 A below a certain input voltage threshold. The desired current curve could be created in Excel or similar tools and exported as CSV file, which could also be loaded on the HMI of the device. The range of the measured voltage on the DC terminal is defined as 0...125% of the rated value, but bringing more than 110% voltage on the device would cause it to switch off, so the effective table value range is 110% or 4096/125*11 = 3605 values. It means that all table values above entry 3605 actually have no effect and could remain 0 or wouldn't be loaded at all. The example below only loads 3277 values, equaling 0...100%.

There is furthermore a global power limitation, so most device models can't make 100% voltage at 100% current. When creating the table in Excel or similar, it might help to add another two columns which, of course, are later not exported to CSV. One column where the referenced value is distributed between 0....110% voltage over 3605 entries and another where the power is calculated for every entry (P = U * I), to find out which of the entries could not be physically realized by the device.

Caution! It's not advisable to have big value differences between two or a group of consecutive table entries. Rather use values that change voltage/current less abrupt.

No.	Command	Description	Register
1		Select the IU mode for the XY generator, i. e. I = f(U). This will switch to function generator mode.	
	FUNC:GEN:SEL␣IUPS	IU table in source mode of bidirectional device or IU table for electronic loads	856
	FUNC:GEN:SEL␣IUEL	IU table in sink mode of bidirectional devices	856
2	**FUNC:GEN:XY:LEV␣0**	Select table entry 0 for writing	2600
3	**FUNC:GEN:XY:DAT␣0**	Write a current value to the table entry, here: 0 A (random value)	2600
...			
6654	**FUNC:GEN:XY:3276**	Select table entry 3276 for writing	5876
6655	**FUNC:GEN:XY:DAT␣120**	Write a current value to table entry 3276, here: 120 A (example)	5876
6656	**FUNC:GEN:XY:SUBM**	Submit all data	

Now it's advisable to also define the set values which are not altered by the table, else the function could run without any effect. It means, if you load an IU table, the current is fetched from the values in the table, but voltage and power are static and could have any value from a previous adjustment.

For an IU table, voltage and power are static. You can set the static values to any value you like, but in order for the static set values not to interfere in the UI or IU function run, it's recommended to set both to maximum:

No.	Command	Description	Register
6657	**VOLT␣0** or **VOLT␣MAX**	With bidirectional devices in sink mode or electronic loads, set the voltage to 0. Alternatively, with power supplies in source mode, set the voltage to maximum or another level that suits the application.	500
6658	**POW␣MAX**	Power to max, independent from the device model	502
6659	Additionally for bidir. series:	Only required for **IU** and **IUEL** modes where the bidirectional device either only works in sink mode or could enter sink mode	
	SINK:CURR␣MAX		499
	SINK:POW␣MAX		498

After this, the function generator is configured and the IU table is loaded. Now the function can be started by remotely controlling the generator:

No.	Command	Description	Register
8197	**INP␣ON** **OUTP␣ON**	Switch the DC input or DC output of your device on	405
8198	**INP␣OFF** **OUTP␣OFF**	Later: switch the DC input/terminal of your device off to make the function stop	405
8199	**FUNC:GEN:SEL␣NONE**	Leave function generator mode with **NONE**	856

5.5.2.3 Command sequence to generate a rising ramp (arbitrary generator)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	✓

Before you can configure the arbitrary generator for a ramp it's necessary to think about the best way to achieve the ramp generation. It's important to keep in mind that the arbitrary generator stops at the end of the function run, unless you configure repetition. After a stop, the DC output remains switched on. In case of a ramp, this is wanted, because the end value shall usually remain set for some time. However, the device will go to static mode again, setting the static set values of U, I and P. The static values also apply for the period before the function run and when the DC output is already switched on.

The stop action and the static values are hence a little problematic for the ramp function. Why? Assuming you wanted to have a power supply generate a ramp starting from 0 V. The static value for U (voltage) would then be set to 0. But after the function stop, the device would also set 0 V and the voltage would drop from whatever value has been reached at the end of the function run. Conclusion: the static value of voltage has to be part of the function setup. In order to achieve this, the function has to consist of two parts: one for the rising or falling ramp and the other for the static value. This can be done using two sequence points of the arbitrary generator.

Assumption: the ramp shall start from 0 V and rise to 50 V within 6 seconds. The end voltage shall remain constant for 3 minutes (the time can be varied at will), sequence points 1 and 2 are used. Remote control is already active, only configuration of the function is necessary. Since the ramp will make the voltage rise linearly, using only the DC part of a sequence point suffices while the parameters for the AC part (indexes 0 - 4) should probably be set to zero in order to avoid previously written values disturbing the wave generation. Particular commands which would achieve this are not included in this example.

Sequence point 1, the rising ramp

No.	Command	Description	Register
1	**FUNC:GEN:SEL␣VOLT**	Selects arbitrary generator for voltage. By sending this command the device will switch to function generator mode	851
2	**FUNC:GEN:WAVE:LEV␣1**	Select sequence point 1 for writing	900
3	**FUNC:GEN:WAVE:IND␣5**	Select index 5: Start voltage of the ramp	
4	**FUNC:GEN:WAVE:DAT␣0**	Set start voltage to 0 V	900
5	**FUNC:GEN:WAVE:IND␣6**	Select index 6: End voltage of the ramp	
6	**FUNC:GEN:WAVE:DAT␣50**	Set end voltage to 50 V	900
7	**FUNC:GEN:WAVE:IND␣7**	Select index 7: Ramp duration	
8	**FUNC:GEN:WAVE:DAT␣6**	Set 6 seconds	900

Sequence point 2, the static voltage at the ramp end

No.	Command	Description	Register
9	**FUNC:GEN:WAVE:LEV␣2**	Select sequence point 2 for writing	915
10	**FUNC:GEN:WAVE:IND␣5**	Select index 5: Start value of the static voltage,	
11	**FUNC:GEN:WAVE:DAT␣50**	Set to 50 V (must be identical to the end value of sequence point 1)	915
12	**FUNC:GEN:WAVE:IND␣6**	Select index 6: End value of the static voltage	
13	**FUNC:GEN:WAVE:DAT␣50**	Set to 50 V	915
14	**FUNC:GEN:WAVE:IND␣7**	Select index 7: Duration	
15	**FUNC:GEN:WAVE:DAT␣180**	Set to 3 minutes (180 seconds)	915

Configuring the arbitrary generator

No.	Command	Description	Register
16	**FUNC:GEN:WAVE:END␣2**	Set sequence point 2 as end sequence	860
17	**FUNC:GEN:WAVE:STAR␣1**	Set sequence point 1 as start sequence	859
18	**FUNC:GEN:WAVE:NUM␣1**	Number of cycles	861
19	**FUNC:GEN:WAVE:SUBM**	Submit all data	

The number of cycles is set to 1, so the function runs once and then stops. It could also repeat, but with every repetition after the 3 m 6 s, the voltage of the ramp would have to drop from 50 V to 0 V, where the ramp is defined to start, but it can't drop in zero time. How long it takes to drop to zero primarily depends on the connected load. The resulting ramp could be more or less malformed. In order to avoid that a third sequence could be configured which just gives the voltage some time to drop.

After this the ramp function is fully configured and can be started. If the DC terminal isn't yet switched on, it will be automatically when running the function. Alternatively, switching it on can also be done separately with the corresponding command. Here it's not required to do so, because the function starts from 0 V, but in case a function shall not start at 0, it would be necessary to switch on the DC output first.

No.	Command	Description	Register
20	**FUNC:GEN:WAVE:STAT␣RUN**	Run the function generator	850

Without repetition the function would stop after one run, after the total time defined as 3 m 6 s by the duration(s) of the sequence point(s) and the voltage would drop to zero. If you wanted to have the static value remain set for much longer, you could extended the duration of sequence point 2 or add a further sequence point which just adds time. With one sequence point you can achieve a duration of 10 h, so the static value could remain set for a maximum of 98 x 10 h = 980 h.

5.5.3 Programming examples for PV simulation (DIN EN 50530)

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	–	–	✓	–	✓

Further information about this extended PV function can be found in the user manuals of your device, as well as in the standard EN 50530. The user manuals also teach about the connection between simulation mode, input mode and panel technology.

Important: after the start of the simulation the device will calculate the first PV curve table. This takes about 500 ms, so the actual simulation begins ≈ 500 ms after the start.

Overview (an "x" marks a possible combination):

Options / Simulation mode	Input mode ULIK	Input mode MPP
ET	x	x (example 1)
UI	-	x (example 3)
DAY ET	x (example 2)	x
DAY UI	-	x (example 4)

5.5.3.1 Example 1

- Technology: cSi
- Input mode: MPP values
- Simulation mode: Continuous, with adjustable temperature and irradiation
- Recording: activated

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:PHOT:MOD␣ET**	Activate PV simulation mode **ET**	12001
3	**FUNC:PHOT:TECH␣CSI**	Select technology: **cSi**	12016
4	**FUNC:PHOT:IMOD␣MPP**	Select input mode: **MPP**	12017
5	**FUNC:PHOT:STAN:MPP:VOLT␣20**	Set MPP voltage: 20 V	12050
6	**FUNC:PHOT:STAN:MPP:CURR␣5**	Set MPP current: 5 A	12051
7	**FUNC:PHOT:REC:ACT␣ENAB**	Activate data recording	12018
8	**POW␣MAX**	Set global power set value to maximum	502
9	**VOLT 30**	Set the global voltage limit (should be higher than Uoc)	500

- *The standard or start values, as set in steps 5 and 6, are only used to calculate the first PV curve. The MPP values (:STAN:MPP) are linked to the solar panel specs Uoc (:STAN:OCV) and Isc (:STAN:SCC) via the factors FFi and FFu. They overwrite each other. It means, that setting :STAN:MPP:VOLT overwrites the value in :STAN:OCV and vice versa*
- *Voltage and current in the MPP are connected via factors FFi and FFu to the open circuit voltage (Uoc) and the short-circuit current (Isc). Depending on the selected technology, these factors are not adjustable*
- *The first curve after function start is calculated with the default values T = 25°C and E = 1000 W/m² (E = from german "Einstrahlung" -> irradiation)*

Control (also during simulation run)

Nr.	Command	Description	Register
10	**FUNC:PHOT:STAT␣RUN**	Start simulation	12000
11	**FUNC:PHOT:TEMP␣40**	Adjust temperature value: 40 °C	12052
12	**FUNC:PHOT:IRR␣800**	Adjust irradiation: 800 W/m²	12053
13	**FUNC:PHOT:STAT␣STOP**	Stop simulation after an arbitrary time	12000

Analysis after simulation end

Nr.	Command	Description	Register
14	**FUNC:PHOT:REC:NUMB?**	Read number (n) of recorded data sets	12020
15	**FUNC:PHOT:REC:IND␣1**	Select first data set (index 1) for reading	12022
16	**FUNC:PHOT:REC:DAT?**	Read data from data set (index) 1	12024
...	**...**	Read further n-1 data sets:	...
x	**FUNC:PHOT:REC:IND␣n**	Select data set n (index n) for reading	12022
y	**FUNC:PHOT:REC:DAT?**	Read data from data set (index) n	12024

5.5.3.2 Example 2

- Technology: Manual
- Input mode: Open circuit voltage and short-circuit current
- Simulation mode: Day trend with adjustable temperature and irradiation
- Interpolation: deactivated
- Data recording: activated

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:PHOT:MOD␣DAYET**	Activate PV simulation mode **DAY ET**	12001
3	**FUNC:PHOT:TECH␣MAN**	Select technology: Manual (all required parameters must be defined, here as with commands 4-10)	12016
4	**FUNC:PHOT:FACT:FFU␣0.8**	Fill factor voltage ($\mathbf{FF_U}$): 0,8	12034
5	**FUNC:PHOT:FACT:FFI␣0.78**	Fill factor current ($\mathbf{FF_I}$): 0,78	12036
6	**FUNC:PHOT:FACT:ALPH␣0.0003**	Temperature coefficient **α** for I_{SC}: 0,0003 /°C	12038
7	**FUNC:PHOT:FACT:BETA␣-0.003**	Temperature coefficient **β** for U_{OC}: -0,003 /°C	12040
8	**FUNC:PHOT:FACT:CU␣0.0725**	Scaling factor $\mathbf{C_U}$ for U_{OC}: 0,0725	12042
9	**FUNC:PHOT:FACT:CR␣0.00022**	Scaling factor $\mathbf{C_R}$ for U_{OC}: 0,00022 m²/W	12044
10	**FUNC:PHOT:FACT:CG␣0.00315**	Scaling factor $\mathbf{C_G}$ for U_{OC}: 0,00315 W/m²	12046
11	**FUNC:PHOT:IMOD␣ULIK**	Select input mode: **ULIK**	12017
12	**FUNC:PHOT:STAN:OCV␣38**	Set open circuit voltage: 38 V	12048
13	**FUNC:PHOT:STAN:SCC␣7**	Set short-circuit current: 7 A	12049
14	**FUNC:PHOT:REC:ACT␣ENAB**	Activate data recording	12018

Nr.	Command	Description	Register
15	**FUNC:PHOT:DAY:INT␣OFF**	Deactivate interpolation of day trend data	12005
16	**POW␣MAX**	Set global power set value to maximum	502
17	**VOLT 38**	Set the global voltage limit (should be ≥Uoc)	500

- *The standard or start values, as set in steps 5 and 6, are only used to calculate the first PV curve. Every change of parameter that would shift the MPP causes the device to calculate the PV anew.*
- *Voltage and current in the MPP are connected via factors FFi and FFu to the open circuit voltage (Uoc) and the short-circuit current (Isc), which are both given in this example, other than in Example 1. Depending on the selected technology, these factors are not adjustable*

Write day trend data (only possible before the function start)

Nr.	Command	Description	Register
18	**FUNC:PHOT:DAY:MOD␣WRIT**	Select access mode: write	12006
19	**FUNC:PHOT:DAY␣CLE**	Delete former data (should be executed every time before loading new data)	12007
20	**FUNC:PHOT:DAY:DAT␣1,␣500, ␣20,␣1500**	Write 1st day trend data set into index 1: Irradiation: 500 W/m² Temperature: 20°C Dwell time: 1500 ms	12010
	The dwell time is defined to have a minimum of 500 ms. However, when setting up the very first day trend data set it's expected to set 1000 ms or higher for this one, because else the function run might fail.		
21	**FUNC:PHOT:DAY:DAT␣2,␣800, ␣28,␣1500**	Write 2nd day trend data set into index 2: Irradiation: 800 W/m² Temperature: 28°C Dwell time: 1500 ms	12010
...	**...**	Write further data sets, a total of 500	...
519	**FUNC:PHOT:DAY:DAT␣500,␣ 1200,␣35,␣20000**	Write 500th day trend data set into index 500: Irradiation: 1200 W/m² Temperature: 35°C Dwell time: 20000 ms	12010

Control

Nr.	Command	Description	Register
520	**FUNC:PHOT:STAT␣RUN**	Start simulation -> the simulation will stop automatically after the time that results from the total of dwell times in all written data sets	12000

Analysis (after simulation end)

Nr.	Command	Description	Register
521	**FUNC:PHOT:REC:NUMB?**	Read number (n) of recorded data sets. This number isn't related to the number of day trend data sets in use. This feature records a new data set every 100 ms. Depending on the total simulation time, the record buffer could fill (max. 16 h record time) and overwrite existing data. It may become necessary to calculate the total simulation time from the day trend data sets and start reading the recorded data during simulation, then clearing the buffer and later read the rest of data.	12020
522	**FUNC:PHOT:REC:IND␣1**	Select first data set (index 1) for reading	12022
523	**FUNC:PHOT:REC:DAT?**	Read data from data set (index) 1	12024
...	**...**	Read further n-1 data sets:	...
x	**FUNC:PHOT:REC:IND␣n**	Select data set n (index n) for reading	12022
y	**FUNC:PHOT:REC:DAT?**	Read data from data set (index) n	12024

5.5.3.3 Example 3

- Technology: thin film
- Input mode: MPP values
- Simulation mode: Continuous, with adjustable MPP (voltage and current)
- Recording: deactivated

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:PHOT:MOD␣UI**	Activate PV simulation mode **UI**	12001
3	**FUNC:PHOT:TECH␣THIN**	Select technology: **Thin film**	12016
4	**FUNC:PHOT:IMOD␣MPP**	Select input mode: **MPP**	12017
5	**FUNC:PHOT:STAN:MPP:VOLT␣45**	Set MPP voltage: 45 V	12050
6	**FUNC:PHOT:STAN:MPP:CURR␣10**	Set MPP current: 10 A	12051
7	**FUNC:PHOT:REC:ACT␣DIS**	Deactivate data recording	12018
8	**POW␣MAX**	Set global power set value to maximum	502
9	**VOLT 57**	Set the global voltage limit (should be ≥Uoc)	500

Control (also during simulation run)

Nr.	Command	Description	Register
10	**FUNC:PHOT:STAT␣RUN**	Start simulation	12000
11	**FUNC:PHOT:STAN:MPP:VOLT␣40**	Shift MPP: 40 V	12050
12	**FUNC:PHOT:STAN:MPP:CURR␣9**	Shift MPP: 9 A	12051
13	**FUNC:PHOT:STAT␣STOP**	Stop simulation after an arbitrary time	12000

5.5.3.4 Example 4

- Technology: cSi
- Input mode: MPP values
- Simulation mode: Day trend with shiftable MPP (voltage and current)
- Interpolation: activated
- Data recording: deactivated

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:PHOT:MOD␣DAYUI**	Activate PV simulation mode **DAY UI**	12001
3	**FUNC:PHOT:TECH␣CSI**	Select technology: **cSi**	12016
4	**FUNC:PHOT:IMOD␣MPP**	Select input mode: **MPP**	12017
5	**FUNC:PHOT:STAN:MPP:VOLT␣36**	Set open circuit voltage: 36 V	12050
6	**FUNC:PHOT:STAN:MPP:CURR␣12**	Set short-circuit current: 12 A	12051
7	**FUNC:PHOT:REC:ACT␣DIS**	Deactivate data recording	12018
8	**FUNC:PHOT:DAY:INT␣ON**	Activate interpolation of day trend data	12005
9	**POW␣MAX**	Set global power set value to maximum	502
10	**VOLT 57**	Set the global voltage limit (should be ≥Uoc)	500

Load day trend data (only possible before the function start)

Nr.	Command	Description	Register
11	**FUNC:PHOT:DAY:MOD␣WRIT**	Select access mode: write	12006
12	**FUNC:PHOT:DAY␣CLE**	Delete former data (should be executed every time before loading new data)	12007

Nr.	Command	Description	Register
13	**FUNC:PHOT:DAY:DAT␣1,␣1,␣1, ␣300000**	Write 1st day trend data set into index 1: MPP voltage: 1 V MPP current: 1 A Dwell time: 300,000 milliseconds => 5 minutes	12010
14	**FUNC:PHOT:DAY:DAT␣2,␣2,␣2, ␣500**	Write 2nd day trend data set into index 2: MPP voltage: 2 V MPP current: 2 A	12010
...	**...**	Write further data set, a total of 1000	...
1012	**FUNC:PHOT:DAY:DAT␣1000,␣30, ␣9,␣500**	Write 1000th day trend data set into index 1000: MPP voltage: 30 V MPP current: 9 A	12010

Due to the dwell time of 5 minutes in the very first day trend data set, all 1000 data sets use the same dwell time, so the total simulation time results as 5000 minutes.

Control

Nr.	Command	Description	Register
1013	**FUNC:PHOT:STAT␣RUN**	Start simulation -> the simulation will stop automatically after the time that results from the total of dwell times in all written data sets	12000

5.5.4 Programming examples for MPP tracking

BT	ELR1	PS1	PSB1	PSBE1	PSI1
✓	✓	–	✓	–	–

5.5.4.1 MPP2

For MPP tracking related information also refer to the user manual of your device. It explains how the different modes work and what the parameters are. Modes 1 to 3 don't require to load user data.

The parameters, as set in the configuration part, are invariable during runtime.

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:GEN:MPP:IND␣0**	Index 0 is used to activate MPP mode selection	11000
3	**FUNC:GEN:MPP:DAT␣MPP2**	Select mode **MPP2** ("track"). This mode doesn't stop automatically.	11000
4	**FUNC:GEN:MPP:IND␣1**	Index 1 = Select to set the open circuit voltage U_{OC} of the solar panel to which the device is connected. This value also defines the voltage limit of the device.	11001
5	**FUNC:GEN:MPP:DAT␣50**	Set U_{OC} to 50 V.	11001
6	**FUNC:GEN:MPP:IND␣2**	Index 2 = Select to set the short-circuit current I_{SC} of the solar panel to which the device is connected. This value also defines the current limit of the device.	11002
7	**FUNC:GEN:MPP:DAT␣100**	Set I_{SC} to 100 A	11002
8	**FUNC:GEN:MPP:IND␣10**	Index 10 = Select to set the tracking interval Δt in milliseconds. This time elapses before the next tracking attempt.	11013
9	**FUNC:GEN:MPP:DAT␣3000**	Time = 3 s	11013
10	**FUNC:GEN:MPP:IND␣6**	Index 6 = Select to set ΔP, an offset to P_{MPP} where the device would start the next tracking attempt when this offset has been exceeded	11006
11	**FUNC:GEN:MPP:DAT␣30**	Set ΔP to 30 W (series with a low power rating have an adjustment range of 0-50 W [see the device's user manual for the actual range], other series have 0-P_{Nom})	11006

Control

Nr.	Command	Description	Register
12	**FUNC:GEN:MPP:STAT␣RUN**	Start the tracking -> the device will attempt to find the MPP and then track it until stopped, because this mode doesn't stop automatically. The time difference between two tracking attempts is defined by index 10, the max. deviation of the actual power from the MPP, before the tracking would continue, is defined via index 6. The last successful action to track the MPP stores three values U_{MPP}, I_{MPP} and P_{MPP} which can be read during runtime or later.	11010
13	**FUNC:GEN:MPP:STAT␣STOP**	Stop tracking anytime	11010

Analysis

Nr.	Command	Description	Register
14	**FUNC:GEN:MPP:IND␣7**	Index 7 = Set read mode for MPP data	11007 11008 11009
13	**FUNC:GEN:MPP:DAT?**	Read the MPP values U_{MPP}, I_{MPP} and P_{MPP}	

5.5.4.2 MPP4

This mode is different to the others. It's available in remote control for all series, but not all of them support to use MPP4 on the control panel (HMI), so this mode isn't explained in these series' user manual. In case you have such a device and need more information, refer to the user manual of series EL 9000 B 3U.

With this mode, also called "user curve", the device runs through 1-100 arbitrarily definable points (voltage values) on a user defined curve, that could represent a PV curve of a solar panel. The goal is to collect measured data and to find the MPP amongst that data. For every processed curve point the device will present readable data in form of MPP values (U, I, P). The actual MPP, derived from all collected data, is also presented as a readable set of data.

In the example it shows how to configure and load 75 user defined points for MPP4 mode.

Configuration

Nr.	Command	Description	Register
1	**SYST:LOCK␣ON**	Activate remote control	402
2	**FUNC:GEN:MPP:IND␣0**	Index 0 is used to activate MPP mode selection	11000
3	**FUNC:GEN:MPP:DAT␣MPP4**	Select mode **MPP4** ("user curve"). This mode will stop automatically.	11000
4	**FUNC:GEN:MPP:IND␣8**	Index 1 = Activate user data input mode	11100 - 11174
5	**FUNC:GEN:MPP:LEV␣1**	Level 1 = 1st point of user curve	
6	**FUNC:GEN:MPP:DAT␣100**	Set 1st point to 100 V	
...		Set further, ideally subsequent points	
153	**FUNC:GEN:MPP:LEV␣75**	Level 75 = 75th point of user curve	
154	**FUNC:GEN:MPP:DAT␣80**	Set 75th point to 80 V	
155	**FUNC:GEN:MPP:IND␣12**	Index 12 = Select to define the range of points to run through in the test (here: end point). The end point can be any of 100 available points. Since the start point can't be higher than the end point, the end point is set first.	11015
156	**FUNC:GEN:MPP:DAT␣75**	End point = 75	11015
157	**FUNC:GEN:MPP:IND␣11**	Index 11 = Select to define the range of points to run through in the test (here: start point). The start point can be any of 100 available points, but must not be higher than the end point.	11014
158	**FUNC:GEN:MPP:DAT␣1**	Start point = 1	11014
159	**FUNC:GEN:MPP:IND␣10**	Index 10 = Select to set the tracking interval Δt, in milliseconds. Defines the time to elapse before proceeding to the next point.	11013
160	**FUNC:GEN:MPP:DAT␣500**	Time = 0.5 s	11013
161	**FUNC:GEN:MPP:IND␣13**	Index 13 = Select to set the repetitions (0-65535) of the curve run. The result data, which can be read later, only holds the results of the last run.	11016
162	**FUNC:GEN:MPP:DAT␣0**	Repetitions = 0 (only one cycle)	11016

Any point that hasn't been loaded and that is within the defined range of points to run through is processed with 0 V.

Control

Nr.	Command	Description	Register
163	**FUNC:GEN:MPP:STAT␣RUN**	Start tracking -> the device will pick the first point of the defined range, set the voltage and measures current and power, stores all three and continues with the next etc. After all points and all cycles are through, the data is scanned for the point with the highest power, which is separately presented as MPP.	11010

Analysis

Nr.	Command	Description	Register
164	**FUNC:GEN:MPP:IND␣9**	Index 9 = Set mode to read the MPP4 measurings	11200 - 11274
165	**FUNC:GEN:MPP:LEV␣1**	Select 1st point to read from the list of measurings	
166	**FUNC:GEN:MPP:DAT?**	Read 1st point as U_{MPP}, I_{MPP} and P_{MPP}	
...		Continuously read further points	
313	**FUNC:GEN:MPP:LEV␣75**	Select 75th point to read from the list of measurings	
314	**FUNC:GEN:MPP:DAT?**	Read 75th point as U_{MPP}, I_{MPP} and P_{MPP}	
315	**FUNC:GEN:MPP:IND␣7**	Index 7 = Set read mode for MPP data	11007 11008 11009
316	**FUNC:GEN:MPP:DAT?**	Read the MPP values U_{MPP}, I_{MPP} and P_{MPP}	

6. Profibus & Profinet

6.1 General

Connection to the field buses Profibus and Profinet are possible via the interface modules I**F-AB-PBUS** (Profibus) or **IF-AB-PNET** (Profinet, 1 or 2 ports) and thus limited to select series:

- all 10000s series

On the device side, the interface module configuration is reduced to the absolute necessary. When using Profibus, the user only has to set a slave address (0...125), while Profinet's network settings are usually configured from remote, best by using the Siemens Primary Setup Tool (PST). Other, optional parameters like tags can be defined in the device's setup menu or via command.

This part of the document shall only teach how to use the so-called ModBus register lists (PDF) with your device in order to access indexes and slots. The list should be enclosed with this document. The interface modules represent the device as a **DP-V1 slave** to the network, capable of cyclic and noncyclical data transmission, whereas the noncyclical one is primary.

6.2 Preparation

For the implementation of a device into a Profibus or Profinet and the enumeration at the master (PLC or similar), a fully configured and wired unit is presumed. The next thing you will usually need is a device description file called GSD (Generic Station Device) for Profibus or a GSDML for Profinet, which is either delivered with the device on the included USB stick or is available as download from the manufacturer's web site or can be obtained upon request.

This GSD/GSDML file enables a specific slot configuration for cyclic process data to be configured, such as actual values or status. Those slots are also used to access other data objects of the device via noncyclical read/write. See more below.

The hardware configuration for a Profibus or Profinet network is originally meant for physical slots in which hardware extensions for the PLC are plugged and addressed. A power supply or electronic load is considered as peripheral hardware with several software slots. The slot model is used to disseminate the many remote control feature across several slots in order to have convenient access. The addressing works the same way as if there were physical hardware extensions plugged into the device, like a digital I/O port.

6.3 Slot configuration for Profibus

The slot configuration for the interface module **IF-AB-PBUS** is done by loading the GSD/GSE file into the hardware catalog, which is usually accessible in the HWCONFIG (Siemens STEP7). The slots must be configured as shown below, even though only slots 1-4 are for cyclic data (DP-V0). The conclusion here is, that only a few data are transferred automatically and only in one direction (device -> PLC) and most of the control objects go via DP-V1, means noncyclical access.

Overview of the slots and their assignment:

Slot	Slot name	Description
1	Device status	Cyclic Device status (for bit layout see register 505)
2	Actual voltage	Cyclic Actual voltage of DC input/output (for translation see *4.4*)
3	Actual. current	Cyclic Actual current of DC input/output (for translation see *4.4*)
4	Actual power	Cyclic Actual power of DC input/output (for translation see *4.4*)
5	Reserved slot 5	not used, but required to configure
...		
8	Reserved slot 8	not used, but required to configure

6.4 Slot configuration for Profinet

The GSDML (available on the included USB stick or as download) doesn't offer automatic slot configuration. When loading the file the correct version for the Profinet interface module in use, 1 port or 2 port, must be selected. After that, the slot placement can be set up like this for cyclic data:

Slot	Slot name	Description
1	Input 2 words	Cyclic device status (for bit layout see register 505)
2	Input 1 word	Cyclic actual voltage of the DC input/output (for translation see *4.4*)
3	Input 1 word	Cyclic actual current of the DC input/output (for translation see *4.4*)
4	Input 1 word	Cyclic actual power of the DC input/output (for translation see *4.4*)

For noncyclical communication, the slots from 5 are indirectly accessed by using an "Index" value that calculates from the slot number and the ID value, as the example in *"6.6.2.1. Activate/deactivate remote control"* shows. This Index, together with the slot 0 and subslot 1 suffices to address all Profinet supported objects from the register lists.

6.5 Cyclic communication

The Profibus / Profinet slave cyclically transfers process data to certain input addresses of the master, as defined by the user for Profibus or Profinet during slot configuration. Also see sections *"6.3. Slot configuration for Profibus"* or *"6.4. Slot configuration for Profinet"*.

Actual values coming from the device have to be translated to real values according to the formula described in section *"4.4. Translating set values and actual values"*, while any other data are referenced in those so-called register lists which usually should come along with this document. The slot names are partially connected to corresponding registers in the lists. For instance, one slot is named "Actual current". The same name can be found in the register list at position 508. This is also where the register is enabled for use with Profibus/Profinet by having a slot/index number assigned. The same principle applies to registers.

Set values, settable status and most other registers are not transferred cyclically for several reasons. One is the high number of available registers which can't be covered by only 16 available slots and the max. data size per slot.

6.6 Noncyclical communication

6.6.1 General

Noncyclical communication with the target device is done by using **slot numbers**, more precisely via their resultant ID, and **indexes**, which are accessed by system function blocks or functions for read or write. When using the Siemens software STEP7, the SFBs to use here are usually SFB52 and SFB53. In TIA Portal there are noncyclical functions called WRREC and RDREC. Other PLC control softwares offer similar options.

The noncyclical SFBs/functions require an ID, an index and a parameter as input. The parameter can be a status or a set value, translated to a hexadecimal value according to *"4.4. Translating set values and actual values"*.

The register list for your device series has two extra columns for Profibus/Profinet use only. These define slot and index number for a particular command. The necessary parameter is defined in the register lists, also in *"4.4. Translating set values and actual values"*. Rule of thumb:

- **Registers where no slot/index is given are not supported via Profibus and Profinet**

The general procedure to control a device remotely is:

1. **Activate remote control** with the appropriate command (may be denied by the device, see *"3.2. Control locations"*)
2. Control and monitor your device remotely, via cyclic (DP-V0) and/or noncyclical (DP-V1) access.
3. Deactivate, i.e. leave remote control

If you just want to record data by reading values from the device, activation of remote control isn't necessary. You can send query commands to the device at anytime and the device will respond immediately, if its current situation allows to respond at all. After querying something from the device, the function block will put out the data returned from the device to an output buffer for further processing.

The field bus ensures that the command is transmitted to the device, otherwise it will generate an error. However, it can't verify that the device really accepted the command or already has set the desired value. This can only be verified by reading the value from the device and comparing. Whether a value has been truly transferred to the device's DC input/output can't be determined definitely.

6.6.2 Examples for noncyclical access

These examples refer to Siemens' automation software STEP7. For noncyclical access (DP-v1) there were function blocks (SFB52, SFB53) used in older versions of this software, but in newer versions like TIA Portal there are now functions called **RDREC** (read record) and **WRREC** (write record) which do the same and which still have the same input parameters ID, INDEX and MLEN. The three have to be determined, whereas the ID always remains the same for a particular device.

6.6.2.1 Activate/deactivate remote control

Remote control is a device state and not the default one. It has to be activated, i.e. requested by the controlling unit from the device before it can be controlled remotely. Depending on the settings and on the state the device is currently in when trying to switch to remote control, the device may deny the request.

► How to activate or deactivate remote control of your device via Profibus or Profinet

1. Use the register list and find the proper command, here: Register 402 - Remote mode.
2. Find the slot and index values for this register in the dedicated columns, for this example it's slot **2** and index 1. In case Profinet is used and there is already an extra column for the Profinet index in your register list (these register lists are update from time to time), read the value from there or calculate (see step 4).
3. Profibus: For older CPUs like the series 300 read the I/Q address for slot 2 from the slot configuration of the device to have the value for parameter "ID". This would by default be set to 260 by the software. Profinet: For newer CPUs like the 1200s, which by default only support Profinet, the ID is equal to the "Hw_IO" value of slot 2 from the system constants.
4. The value "index" from the register list is submitted to the parameter INDEX as follows:

   Profibus: **INDEX** = index = **1**

   ID = as read, e. g. 260 or DW#16#104

   MLEN = 2 (equals to one ModBus register)

   Profinet: **INDEX** (if calculated) = slot from register list* 255 + 1 + index from register list = **512 (0x200)**

   ID = as read, e. g. 260 or DW#16#104, or as read from "Hw_IO"

   MLEN = 2 (length in bytes, as read from the register list for address 402)
5. Use a suitable function block in your automation software, such as SFB53 / DB53, or function WRREC.
6. Define the control value to use for this command, as described in the columns “Data” and “Example”:

   0xFF00 = Activate remote control

   0x0000 = Deactivate remote control
7. Feed the control value to the function block SFB53 or function WRREC into **REQ**, together with the other three. If not somehow inhibited by the device, it should either switch to remote control or back to manual control.

6.6.3 Send a set value

Any command that sets something in the device, no matter if value or status, requires activated remote control status. Also see *"6.6.2.1. Activate/deactivate remote control"* and *"3.2. Control locations"*.

Before you send a value, you first need to select which one you want to set and you also might need to translate it, because via Profibus/Profinet set values are transferred as per cent of the nominal values. Read sections *"4.3. Format of set values and resolution"* and *"4.4. Translating set values and actual values"* for more information.

► How to set the DC input/output current value

1. Use the register list and find the proper command, here: Register 501 - Set current value.
2. Find the slot and index values for this register in the dedicated columns of the register list, for this example it's slot **2** and index 24. In case Profinet is used and there is already an extra column for the Profinet index in your register list (these register lists are updated from time to time), read the value from there or calculate (see step 4).
3. Profibus: For older CPUs like the series 300 read the I/Q address for slot 2 from the slot configuration of the device to have the value for parameter "ID". This would by default be set to 260 by the software. Profinet: For newer CPUs like the 1200s, which by default only support Profinet, the ID is equal to the "Hw_IO" value of slot 2 from the system constants.
4. The value "index" from the register list is submitted to the parameter INDEX like this:

   Profibus: **INDEX** = index = **24**

   ID = as read, e. g. 260 or DW#16#104

   MLEN = 2 (equals to one ModBus register)

Profinet: **INDEX** (if calculated) = slot from register list* 255 + 1 + index from register list = **535 (0x217)**

ID = as read, e. g. 260 or DW#16#104, or as read from "Hw_IO"

MLEN = 2 (length in bytes, as read from the register list for address 402)

5. Use a suitable function block in your automation software, like SFB53 / DB53, or function WRREC.
6. Define the control value to use for this command, as described in the columns “Data” and “Example”. First, read the value range: 0x0000...0xCCCC (decimal: 52428) = Current 0...100%. Second, calculate the set value.

   For a model with, for example, 170 A nominal current and a desired current of 10 A, this would be 52428/17 = 3084 --> 0x0C0C.
7. Put the control value 0x0C0C together with ID and INDEX into input **REQ** of the function block SFB53 or function WRREC and execute it. The device should instantly set 10 A as current limit. This can be verified in the display of the device where it shows the set value of current.

6.6.4 Read something

Reading something from the device is always possible, it means that no remote control is required. Apart from the cyclically transferred data, any other available information can be read via noncyclical transfer.

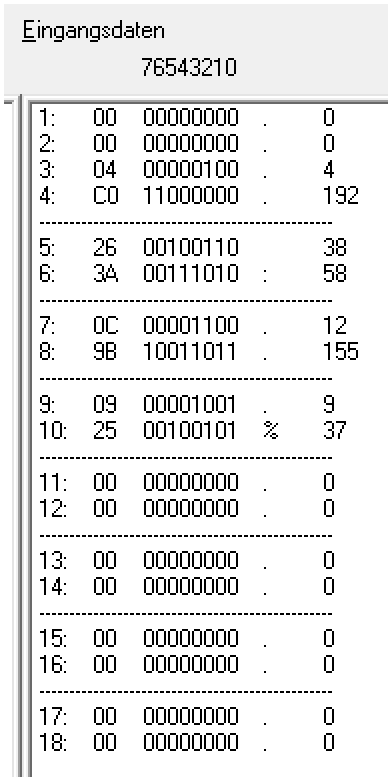
► How to read the actual values of voltage and current

1. Use the register list and find the proper register. The registers of voltage and current are next to each other, the one for voltage has the lower number, so it will be: Register 507 - Actual voltage
2. Determine the values for ID, INDEX and MLEN as shown in the above examples.
3. Read the length of bytes from the column “Data length in bytes” to determine how many bytes to read. In this case there are two registers with length 2 bytes to read, so it's 4 bytes.
4. Configure block SFB52 or function RDREC with ID, INDEX and data length (4 bytes or 2 words of 16 bit, depending on the way the software defines the input).
5. Execute the function block. The data buffer of the block should return the requested data in form of 4 bytes.

The returned 4 bytes will contain the actual voltage value in the first two bytes which is represented as per cent value (for translation see *"4.4. Translating set values and actual values"*). The actual current value will be in the last two bytes. By varying the data length to 6 you could also include the actual power value. Alternatively, you can query each actual value separately. To do this, you need to use the corresponding register number to calculate the INDEX and a data length of 2.

6.7 Data interpretation

Data returned from queries, but cyclically transferred data in the first place, have to be interpreted. Let's use an example from a Profibus master simulator where the cyclic data is comfortably displayed. Also see section *"4.4. Translating set values and actual values"*.



```
Eingangsdaten
          76543210

1:   00  00000000  .    0
2:   00  00000000  .    0
3:   04  00000100  .    4
4:   C0  11000000  .    192
-----------------------------
5:   26  00100110       38
6:   3A  00111010  :    58
-----------------------------
7:   0C  00001100  .    12
8:   9B  10011011  .    155
-----------------------------
9:   09  00001001  .    9
10:  25  00100101  %    37
-----------------------------
11:  00  00000000  .    0
12:  00  00000000  .    0
-----------------------------
13:  00  00000000  .    0
14:  00  00000000  .    0
-----------------------------
15:  00  00000000  .    0
16:  00  00000000  .    0
-----------------------------
17:  00  00000000  .    0
18:  00  00000000  .    0
```

The figure shows the transferred data of a configuration with 8 slots. Because only slots 1-4 are used for cyclic transfer, the rest remains empty.

Slot 1: Device status (connected to register 505). The exemplary value 0x000004C0 tells that bits 6, 7 and 10 are set, meaning the device is configured as master (for master-slave), the input/output is on and regulation mode is CC.

Slot 2: Actual voltage (connected to register 507). With a 250 V model, for instance, the value 0x263A translates to 250 V*0x263A/52428=46.7 V.

Slot 3: Actual current (connected to register 508). With a 510 A model, for instance, the value 0x0C9B translates to 510 A*0xC9B/52428=31.4 A.

Slot 4: Actual power (connected to register 509). For a 5 kW power supply, for instance, the value 0x0925 translates to 5000 W*0x925/52428=223 W or 0.22 kW.

Slot 5: not used for cyclic data

Slot 6: not used for cyclic data

Slot 7: not used for cyclic data

Slot 8: not used for cyclic data

For users of a PSB 9000 or PSB 10000 device: same as represented on the device's display, actual values of current or power can be negative, depending on the operation mode. The definition of those registers holding the actual value doesn't support negative values, but the "Device status" register 505 (cyclic or noncyclical readable) indicates the operation mode via bit 12, i. e. source or sink mode. The bit can be used to invert the actual values of current and/or power for sink mode and further processing.

7. CANopen

The available communication objects (ADIs) are defined in an Electronic Data Sheet file (EDS/XDD), which is delivered with your device on a USB stick or are available as download from the manufacturer's website. This EDS can be integrated in CANopen related software. The indexes in the EDS are not separately explained, because their definition and use is identical to the herein described ModBus protocol and the related, external register list files (see *"4.6. About the register lists"*). Examples from the ModBus part of this document can be used and applied for CANopen as well, but would be reduced to the core data, because CANopen user are not confronted with checksums and function codes as with ModBus.

The CANopen module IF-AB-CANO doesn't feature an internal termination resistor, so the bus termination has to be applied by the user according to the CAN bus requirements.

7.1 Restrictions

- Internally, the device handles everything based on ModBus with a register set that starts at address 0. With the 10000 series and with the CANopen interface, the device's object list is placed in the manufacturer specific object range which starts at index 0x2001. That shifts the ModBus register addresses for CANopen by the value 0x2001, but at the same time keeps their order as listed in the register lists. Address 0x5FFF is the highest addressable object in this range, corresponding to ModBus register 16383. It means, with CANopen it's not possible to use all the features of a device, specifically those beyond that register number.
- No PDO mapping possible
- No heartbeat, no guarding

7.2 Preparation

For the communication with the device via CANopen interface **IF-AB-CANO**, a few things are required:

1. A suitable CAN cable, preferably with switchable termination resistor, which has to be always activated if the device is at the end of the bus, like when directly connecting the PC to a single ELR 10000 unit.
2. EDS/XDD (included with the device on USB stick).
3. CANopen software for the PC (not included with the device, any available software for CANopen should suffice).
4. Documentation about how to use the supported indexes. See sections *1. - 4.*, *7.2* and *9.*, as well as the included register list(s).

7.3 User objects

The data format used via CANopen communication is related to ModBus. A specific object index is connected to a specific ModBus register. The CANopen standard defines that user objects are enumerated from index 2001, which a hexadecimal number without the usual prefix 0x. With ModBus, the registers are counted from 0. It means, that index 2001 corresponds to register 0 or index 21F5 corresponds to register 500 etc.

The EDS/XDD contains less objects than the device supports as ModBus registers, but the available objects still cover the most important functions of the device. Users can edit the EDS/XDD anytime and add further objects.

Together with this document there are usually so-called register lists for primary ModBus use, but these can also be used for CANopen, as they also define data type and value range of the objects. Control examples in other sections of this document can be applied for CANopen as well.

7.3.1 Translation CANopen index to ModBus register

The translation of a CANopen index, as listed in the EDS file, to a ModBus register address is quite easy due to the fixed offset 0x2001. For example, if you pick the object index "207A Nominal voltage" from the EDS, it translates like this:

Index number - Offset = register address --> 0x207A - 0x2001 = 0x79 (hex) = 121 (dez). According to any register list this represents the rated device voltage as a FLOAT value. Because CANopen doesn't support the data type FLOAT, the EDS uses REAL32 here, which is the equivalent of a single-precision float value. The user just has to translate the 32 bit value according to IEEE 754 specification.

7.4 Specific examples

7.4.1 Switching to remote control

As described in *"4.7.8.5. Switch to remote control or back to manual control"*, it's required to switch the device to remote control before you can control it. In order to do this, you first need to find the proper command, i.e. register in the register list or the dedicated index in the EDS. In this case, it's register 402 or index 0x2193. The register list defines that the value 0xFF00 has to be sent to switch to remote or value 0x0000 to leave remote control. So a value of 0xFF00 would be written by SDO transfer to index 0x2193.

7.4.2 Setting a set value

After remote control has been accepted by the device, you are allowed to send set values. Those values usually represent a per cent value of the corresponding rated value. From the definition in the register list, 100% of a value translates to the hexadecimal value 0xCCCC and 0% to 0x0000. It means, there are 52429 possible values between 0% and 100%. It has to be pointed out here, that this isn't the true resolution values like voltage or current actually achieve at the DC input/terminal. The effective resolution of DC related values is 26214 steps. An example for set value translation is in *"4.7.8.1. Writing a set value"*.

7.4.3 Using the arbitrary generator

Due to the nature of CAN and thus also CANopen, all 8 parameters of a sequence point cannot be written at once. With date 01-2024, the 20000 series devices feature CAN FD and BRS and with the max. data length of 64 bytes a complete sequence point could be written in one frame, but this isn't yet supported.

When writing the actually used parameters (not all of them have to be written every time) as subindexes, you would usually send them from the first to the last. However, when writing any sequence point for the first time after the device has been started, the value for parameter Time in the memory of the device's controller might be 0, because not correctly initialized in some firmwares. Therefore, writing any other parameter than time as the first one might result in an error, because the device checks them upon reception against a minimum slope requirement (see section *"3.9. Minimum frequency slope (arbitrary function generator)"*) and comparing to 0 generates the error. Simple solution: always write the subindex 8 for the Time parameter first.

Once all sequence points are set without any error **it requires to send an additional submit command (index 235F)**. This will transfer all sequence point data and load the function into the function generator and enable start/stop action. Without sending that command the function generator would either run with all data being zero or using old data.

The steps to perform for CANopen are the same as in the example in section *4.10.8.1* which is for direct ModBus communication when using a different interface:

Step 1:

Select, whether to apply the function to the voltage U (index 2354) or the current I (index 2355). Before this selection isn't done, the device can't accept sequence point data, because the data is run through a plausibility check against the device’s rated values.

Step 2:

Define start sequence point (index 235C), end sequence point (index 235D) and number of cycles the block of sequence points shall be repeated (index 235E).

Step 3:

Load data for all required sequence points (x out of 99, indexes 2385 - 29A5, 8 values per sequence point in sub indexes).

Step 3.1:

Submit the data by writing 0xFF00 to index 235F (register 862).

Step 4:

Set the global voltage limit (index 21F5), if the function is applied to the current. Otherwise, set the global current limit (index 21F6), if the function is applied to voltage. Set the global power limit (index 21F7) for both modes. When setting this up for a bidirectional device where the function might even switch between sink and source mode, two more global values are required to be set. These the sink current (index 21F4), in case the function is applied to voltage, and the sink power (index 21F3). Both are either set to the desired maximum or to 0 if the device shall only run in source mode.

Step 5:

Control the function generator with start/stop (index 2353).

Step 6:

As soon as the function generator isn't needed anymore, leave function generator mode by deselecting your former selection from step 1, either U (index 2354) or I (index 2355) by writing 0x0000.

7.5 SDO abort codes

Following SDO transfer abort/error codes, as part of the CANopen standard, are supported by the CANopen interface module IF-AB-CANO:

Code	Description
0x06020000	Object doesn't exist in the object dictionary (ModBus register list)
0x06040043	Command not supported
0x06099911	Sub-Index doesn't exist
0x06010002	Attempt to write a read only object
0x06010002	Attempt to read a write only object
0x06070012	Too much data
0x06070013	Not enough data
0x06090030	Value range of parameter exceeded
0x08000022	Data could not be transferred or stored to the application because of the present device state
0x05040005	Out of memory
0x08000000	General error

8. CAN / CAN FD

This section is dedicated to the communication with a device via the CAN interface IF-AB-CAN (for 10000 series) or the built-in CAN/CAN FD interface as available with the 20000 series. Configuration of the interface itself is done on the control panel (HMI) of the device or via remote control for those series without display. The single settings are described in the device manual.

8.1 Preparation

What is required for the communication with the device via CAN?

1. A suitable CAN cable. It's not required to have one with integrated bus termination switch and resistor, because the interface module IF-AB-CAN and the integrated CAN port on the 20000 series device both feature an electronically switched resistor for bus termination. In case the cable also has one, it's important to activate only the resistor in the cable or the device, otherwise there can be bus errors.
2. When using software from Vector™ company or other software which can make use of so-called database files (DBC), a dedicated DBC for the particular device model is usually required which is delivered with the device on USB stick or can be created by the user completely new or by modifying a similar one. Regarding the 20000 series, there is the choice to work in CAN FD mode or not. The dedicated 20000 series DBCs are already compatible with CAN FD mode and are usually all pre-configured and delivered with the device on USB stick. In case of a multi-channel model, such as from BT 20000 Triple series, all predefined messages in the DBC exist three times, using different Base IDs, as configurable by on the HMI's display or via remote control. The DBC in use always comes with the default Base ID(s) and in case a different range of IDs has been assigned to the device or multiple of our devices run on the same channel, the IDs for the messages in the DBC must be edited.
3. CAN software for the PC (not included, any available software for CAN should suffice).
4. Documentation about how to use the supported CAN objects. See below and sections *1. - 4.*, as well as the included register list(s).

8.2 Introduction

The data format is derived from the previously in this document described ModBus RTU. In relation to a database file (DBC) a **mux value** (Vector terminology) represents a specific **ModBus register**, here also considered as object or command. Objects in the database are selected by the so-called "muxer" or, when programming the CAN message buffer directly (CAPL), the first two bytes of data in a CAN message directly define the ModBus register address that shall be accessed. The selection between writing and reading objects is done by the CAN ID. With multi-channel devices, the CAN ID additionally selects the channel.

Each device will be assigned at least three CAN IDs, which are defined via the so-called Base ID in the device's CAN settings. With multi-channel devices, there will be a dedicated Base ID for every channel. The Base ID of a device or channel is used to write to objects (connected DBC message: **Send_Object**), while querying objects (connected DBC message: **Query_Object**) is done with the ID that calculates from Base ID+1. Responses (connected DBC message: **Read_Object**) coming from the device or channel use an ID calculated from Base ID+2. Such responses are expected after a query, but can also be received unexpectedly in case of communication or access error. When adjusting the Base ID(s) of a device, the other two related IDs will shift automatically.

There is another adjustable ID, the Broadcast ID. It's separate from all others and can be used to address multiple devices or multiple channels of a multi-channel device at once. It only requires to have set this Broadcast ID to the same value on every device/channel where broadcast shall be used. This ID is for write access (Send_Object) only. Queries to multiple devices at once by sending one message are not possible.

Apart from the Base ID and Broadcast ID for noncyclical access, some series support further adjustable IDs for cyclic status data which is sent permanently by the device, when activated, in an adjustable interval and immediately after the CAN connection has been established. Refer to the device manual for CAN setup details, particularly to the section for the communication settings.

8.3 Message formats

8.3.1 Normal sending (writing)

Writing to the device is always done via the Base ID or the broadcast ID of the device or a specific channel (multi-channel devices). It requires to define the first register address to write data to, as well as the number of registers to write and a specific number of parameter bytes which can represent different data types.

Connected ModBus functions: Write Single Coil (WSC), Write Single Register (WSR)

Access via: Base ID, broadcast ID

Used for: set values and similar

Bytes 0+1	Byte 2	Bytes 3+4
Register	**Nr. of regs to write**	**Data**
0...65534	Always 1	Value (16 bit)

Connected ModBus function: Write Multiple Registers (WMR)

Access via: Base ID, broadcast ID

Used for: writing multiple 16 bit or 32 bit values or strings

Bytes 0+1	Byte 2	Byte 3	Bytes 4-7
Start reg.	**Nr. of regs to write**	**Marker**	**Data bytes**
0...65534	2...123	0xFF, 0xFE...	Four bytes or two 16 bit values or one 32 bit value

Start register: always contains the first register address of a register or register block

Nr. of regs to write: refer to the register list. An object defined with 40 bytes occupies 20 registers, so when writing to such an object the value here would have to be 20.

Marker: used to send connected messages where the payload would exceed what can be transported in one CAN message. This marker is also used to define the correct sequence of data. For example, a string like the user text (register 171) can be up to 40 characters and when writing it has to be split across multiple messages. Every message can transport 4 characters of the string. The marker always starts with 0xFF and is counted down (0xFF, 0xFE...) with every next split message belonging to a transmission. The marker is required, because on the CAN bus it's not guaranteed that messages are received in the same order they were sent or at least not as one block of subsequent ones.

Data bytes: the number of data bytes or payload in this type of message is always 4, no matter if all bytes are filled with information or are 0. An example: a user text with a length of 15 characters would require to send 4 messages. The object for the user text is defined to have 20 registers, means 10 messages. Since you would write to less registers than defined you would only have to reduce the number of **Nr. of regs to write**. In this example it would be 8, resulting in 4 messages containing 16 bytes of payload (15 bytes of string + termination character).

8.3.2 Cyclic sending (writing)

Cyclic sending (or writing) is an additional way which allows for compact and time efficient transmission of often used set values and status in form of **grouped data**. Cyclic sending uses extra CAN IDs for a device or every channels of multi-channel devices. How "cyclic" the data is transmitted to the device is determined by the controlling side, which defines the interval via the global message timing. We still recommend to stick to the timing as described in section *3.3.3*.

8.3.2.1 Group part "Control"

Access via: Base ID Cyclic Send

Bytes 0-1
Control word

Control word definition:

Bit	Name	Related register	Meaning
8	Remote control	402	Activates remote control of the device with 1 or deactivates it with 0
9	DC terminal/input	405	Switches the DC input/terminal of the device or addressed channel of a multi-channel device on with 1 or off with 0
10	UIP / UIR	409	Activates resistance control mode (UIR, where featured) with 1, while with 0 mode UIP will be active
11	-	-	-
12	Alarms	411	A "1" acknowledges all currently acknowledgeable alarms

This control word requires special attention, as all bits can trigger several actions at once which don't have a certain priority of processing. It means, if you would try to activate remote control together with switching the DC input/terminal on, i. e. bits 0 and 1 both TRUE, you may receive a settings conflict error, because the device would possibly process bit 9 before bit 8.

8.3.2.2 Group part "Set values 1"

Access via: Base ID Cyclic Send + 1

Bytes 0-1	Bytes 2-3	Bytes 4-5	Bytes 6-7
Register 500	**Register 501**	**Register 502**	**Register 503**
Set value of voltage	Set value of current	Set value of power	Set value of resistance

With bidirectional devices, these four set values belong to the "source mode", shown as "Set values [PS]" on the HMI under "Base ID Cyclic Send".

8.3.2.3 Group part "Set values 2" (only for bidirectional devices)

Access via: Base ID Cyclic Send + 2

Bytes 0-1	Bytes 2-3	Bytes 4-5
Register 499	**Register 498**	**Register 504**
Set value of current (EL)	Set value of power (EL)	Set value of resistance (EL)

With bidirectional devices, these three set values belong to the "sink mode", listed as "Set values [EL]" on the HMI under "Base ID Cyclic Send".

8.3.3 Querying

Querying an object is the first part of a read action. It's always done via Base ID + 1. The device or addressed channel of a multi-channel device should then respond via Base ID + 2 (connected DBC message: Read_Object) and with the expected data. Only after reading the response, the read action is finished. In order to query an object via the Query ID (Base ID +1) it's sufficient to send the start register number. The device will respond with the correct length of data, but in different formats. See below at *8.3.4*.

Connected ModBus functions: Read Coils (RC), Read Holding Registers (RHR)

Access via: Base ID + 1

Bytes 0+1
Start reg.
0...65534

8.3.4 Normal reading

Data coming from the device can be a single message (expected data or error) or can be split messages forming a response to a query. Other messages coming from a device, those which are not queried, are dealt with in *8.3.5*. The information in the message will be put into a buffer and, when using software from Vector company, automatically sorted into signals. The data of split messages has to be combined again and according to the marker. Even the Vector database can't do this automatically. However, there are only a few objects, such as the user text, which require this kind of treatment and they're usually not accessed that often.

Incoming from the device via: Base ID + 2

Response as one message (number of queried registers 1-3):

Bytes 0+1	Bytes 2-7
Register	**Data**
0...65534	1-3 registers

Response as multiple messages (number of queried registers >3):

Bytes 0+1	Byte 2	Bytes 3-7
Register	**Marker**	**Data**
0...65534	0xFF, 0xFE...	5 bytes

Note: Byte 2 can't be used to determine whether the message is part of a split response, as single registers can also return 0xFF in Byte 2.

Response as error message:

Bytes 0+1	Byte 2
65535	Error code

The error codes used here are partially the same as with ModBus, partially CAN specific. See *"8.3.6. CAN communication error codes"*.

8.3.5 Cyclic reading

The cyclic reading is an additional feature where the device or one or several channels of a multi-channel device can automatically send specific objects to user-definable IDs in a user-definable interval. Messages coming in from cyclic reading are different from those of normal "noncyclical" read actions, as described in *8.3.4*. In order to activate and use cyclic reading, the user has to...

1. define the separate "Base ID Cyclic Read" on the device (HMI, CAN settings) or via remote control.
2. define which of the 7 available items for cyclic reading are going to be used and activate them by setting the interval time to a value other than zero. With multi-channel devices, this would have to be considered for every channel separately.
3. process the received data separately, because the data format is different here (see below).

The interval times for the cyclic objects can each be set separately. In case the time settings match or overlap, the device will send the corresponding messages subsequently and as quick as possible.

The minimum interval is 20 ms. When using a very low CAN bus speed, for example 10-50 kbps, CAN bus errors may occur when multiple cyclic reading items are active, because of too much traffic. When communicating with a multi-channel device and when having cyclic reading activated on more than one channel, the traffic will even be higher. In the worst case, the device could be sending 21 messages at once.

Once cyclic reading is activated by setting the interval time of at least one item and as soon as a CAN connection is established, the device will start to automatically and permanently send messages using the defined IDs. The cyclic reading feature can be turned off or on anytime using the CAN settings on the HMI or the corresponding commands being sent as noncyclical CAN messages. There are up to 7 CAN IDs reserved for cyclic reading. Starting at the adjustable "Base ID Cyclic Read" (see HMI of the device) the data in the messages is defined as follows.

8.3.5.1 Message "Status"

Incoming from the device via: Base ID Cyclic Read

Bytes 0-3
Device status (32 Bit)

Bit layout of the device status value:

Bit	Name	Meaning
31	Remote control	1 = on
30	DC input/terminal	1 = on (req., register 405)
29	-	-
28	Operation mode	0 = UIR, 1 = UIP
27	Alarms	1 = at least 1 alarm active
26	Alarm MSP/MAP	1 = alarm active
25	Alarm OCD	1 = alarm active
24	Alarm OCP	1 = alarm active
23	Interface in access	Register 505, bits 4-0
22		
21		
20		
19		
18	Alarm OPD	1 = alarm active
17	Alarm OPP	1 = alarm active
16	Alarm OT	1 = alarm active
15	-	
14	Alarm OVD	1 = alarm active
13	Alarm OVP	1 = alarm active
12	Alarm PF	1 = alarm active
11		
10		
9	REM-SB (1	1 = on (register 505, bit 30)
8	Alarm UCD	1 = alarm active
7	Alarm UVD	1 = alarm active
6	Remote sensing	1 = external, 0 = internal
5	Function gen.	1 = FG active
4	MS type	1 = master, 0 = slave
3	Input / output	1 = on (register 505, bit 7)
2	Regulation mode	Register 505, bits 10-9
1		
0	Bidirectional mode	0 = source, 1 = sink

1) Only with 10000 series devices

8.3.5.2 Message "Actual values

Incoming from the device via: Base ID Cyclic Read + 1

Bytes 0-1	Bytes 2-3	Bytes 4-5
Register 507	**Register 508**	**Register 509**
Actual voltage	Actual current	Actual power

Bidirectional devices only: these actual values are unsigned, so it's not clear whether they belong to source or sink mode. In order to find out, bit 0 of block "Status" would be evaluated.

8.3.5.3 Message "Set values 1"

Incoming from the device via: Base ID Cyclic Read + 2

Bytes 0-1	Bytes 2-3	Bytes 4-5	Bytes 6-7
Register 500	**Register 501**	**Register 502**	**Register 503**
Set value of voltage	Set value of current	Set value of power	Set value of resistance

With bidirectional devices, these four set values belong to source mode operation, listed as "Set values [PS]" on the HMI under "Base ID Cyclic Read".

8.3.5.4 Message "Limits 1"

Incoming from the device via: Base ID Cyclic Read + 3 ("Limits 1" or "Limits 1 [PS]")

Bytes 0-1	Bytes 2-3	Bytes 4-5	Bytes 6-7
Register 9002	**Register 9003**	**Register 9000**	**Register 9001**
I-max	I-min	U-max	U-min

With bidirectional devices, these adjustment limits only belong to source mode operation, listed as "Limits 1 [PS]" on the HMI under "Base ID Cyclic Read".

8.3.5.5 Message "Limits 2"

Incoming from the device via: Base ID Cyclic Read + 4 ("Limits 2" or "Limits 2 [PS]")

Bytes 0-1	Bytes 2-3
Register 9004	**Register 9006**
P-max	R-max

With bidirectional devices, these adjustment limits only belong to source mode operation, listed as "Limits 2 [PS]" on the HMI under "Base ID Cyclic Read".

8.3.5.6 Message "Set values [EL]"

Only available with bidirectional devices.

Incoming from the device via: Base ID Cyclic Read + 5

Bytes 0-1	Bytes 2-3	Bytes 4-5
Register 499	**Register 498**	**Register 504**
Set value of current (EL)	Set value of power (EL)	Set value of resistance (EL)

8.3.5.7 Message "Limits [EL]"

Only available with bidirectional devices.

Incoming from the device via: Base ID Cyclic Read + 6

Bytes 0-1	Bytes 2-3	Bytes 4-5	Bytes 6-7
Register 9008	**Register 9009**	**Register 9005**	**Register 9007**
I-max	I-min	P-max	R-max

8.3.6 CAN communication error codes

Communication errors are returned from the device using the same error codes as those from ModBus communication, but there are additional ones related to the CAN module IF-AB-CAN for the 10000 series, because that module is a gateway and runs its own firmware. Those error codes would also be used by 20000 series devices, if applicable.

For the format of a CAN error message see *"8.3.4. Normal reading"*.

Code	Description
0x02	Invalid register address. Means, the register address given in the CAN message is not specified for the addressed device. See the series specific register list.
0x03	Wrong message or payload length. Depending on the addressed register, a payload of two or more bytes is required. This error would come if the DLC of a CAN message is 5 while it should be 6 or more.
0x04	Execution error. This is a general error caused by a conflict between a correct command and the device's situation. Example: it would be caused when trying to change the mode of the function generator (where featured) while it's already active and running.
0x07	Access denied. An attempt was done to write to a register that is defined as "read only" or vice versa.
0x17	Device in local. Can only occur when trying to enter remote control by a correct command while the remote control is block by "Local" mode being activated for the device. This is actually the same situation as with error 0x04, it's only more specific.
0x20	Overload. Can occur when too many messages arrive in the buffer of the IF-AB-CAN module, so the messages boxes run full. The solution would usually be to reduce the amount of traffic or to increase the time between subsequent messages to the same CAN ID.

8.4 Examples

8.4.1 Switching to remote control

As described in *"4.7.8.5. Switch to remote control or back to manual control"*, it's necessary to switch the device to remote control before you can control it. In order to do this, you first need to find the proper command, i.e. register in the register list. In this case, it's register 402 (hex: 0x192). The register list defines that the value 0xFF00 has to be sent to switch to remote or value 0x0000 to leave remote control. Assuming the device or any channel of a multi-channel model has been set to Base ID 0x20, the data to be sent according to *8.3.1* would be:

$$\underbrace{\boxed{\text{0x01}}\boxed{\text{0x92}}}_{\text{Register / object}}\underset{\text{Nr. of regs}}{\boxed{\text{0x01}}}\underbrace{\boxed{\text{0xFF}}\boxed{\text{0x00}}}_{\text{Bit (coil) for TRUE}}$$

The device should switch to remote control immediately, if not inhibited somehow. The status of remote control can be read from the display or by reading status register 505.

8.4.2 Write and read back a set value

After remote control has been accepted by the device, you may start to send set values. Those values usually represent a per cent value of a rated value. From the definition in the register list, the hexadecimal value 0xCCCC translates to 100% and 0x0000 to 0%. It means, there are 52429 possible values between 0% and 100%. It has to be pointed out here, that this isn't the actual resolution of a voltage or current value at the DC input/terminal. The effective resolution is 26214 steps. An example for set value translation can be found in *"4.7.8.1. Writing a set value"*.

Message example: power supply model PSI 10080-170 3U has a rated current of 170 A. If you wanted to set it to 35 A, the set value would calculate as 35 A * 52428 / 170 A = 10794 = 0x2A2A, according to the formula in *4.4*. The current is set with register 501. Assuming the device would have been set to Base ID **0x88** the data to send to this ID would have to be like this, according to *8.3.1*:

$$\underbrace{\boxed{\text{0x01}}\boxed{\text{0xF5}}}_{\text{Register / object}}\underset{\text{Nr. of regs}}{\boxed{\text{0x01}}}\underbrace{\boxed{\text{0x2A}}\boxed{\text{0x2A}}}_{\text{Bit (coil) for TRUE}}$$

Soon after the device has received and accepted the value, it's set and could be read from the display or also by reading it back using the same object. Given the same base ID, the query message to send using CAN ID **0x89** would be:

$$\underbrace{\boxed{\text{0x01}}\boxed{\text{0xF5}}}_{\text{Register / object}}$$

Short after this, the device should respond the requested value on the read ID **0x8A**:

$$\underbrace{\boxed{\text{0x01}}\boxed{\text{0xF5}}}_{\text{Register / object}}\underbrace{\boxed{\text{0x2A}}\boxed{\text{0x2A}}}_{\text{Set value of current}}$$

In case the value hasn't been accepted, for example when the adjustment limit for current (I-max) would be set to 30 A, the device may also return an error message instead:

$$\underbrace{\boxed{\text{0xFF}}\boxed{\text{0xFF}}}_{\text{Error}}\underset{\text{Error code}}{\boxed{\text{0x03}}}$$

The ModBus error code 0x3 tells "wrong data". Also see *4.9*. In this case, the set value was too high.

8.4.3 Programming the arbitrary function generator

The procedure here is identical to CANopen. Therefore, please refer to *"7.4.3. When using the arbitrary generator"*.

9. EtherCAT

9.1 Preamble

Per definition, the EtherCAT communication with our devices is based on cyclic (PDO) and noncyclical (SDO) objects, whereas the latter used "CoE", i. e. CANopen objects transferred via an Ethernet network connection, also called "CANopen over Ethernet". All documentation for EtherCAT and CANopen is provided by the Beckhoff company or the CiA organisation.

There are differences between devices from the 10000 series and 20000 series regarding the PDO configuration, the object dictionary (OD) indexes and the mapping feature which is only supported by the 20000 series.

9.2 Restrictions

- With 10000 series: a scan for devices on the EtherCAT might not detect a bidirectional device as such from the ESI, if installed, except the ESI has been modified before (more see below) or isn't installed, so the configuration would be read from the slave. The ESI based scan would find a "IF-AB-ECT for standard" where the PDOs wouldn't contain the extra objects needed for a bidirectional device -> the circumvention would be to manually add a slave via dialog (see the example screenshot from Beckhoff's TwinCAT below) and device type selection.
- No hot-plug support; devices being powered while the network is already running must be switched to "OP" mode separately
- No support for object mapping in the PDOs

9.3 Requirements and preparation

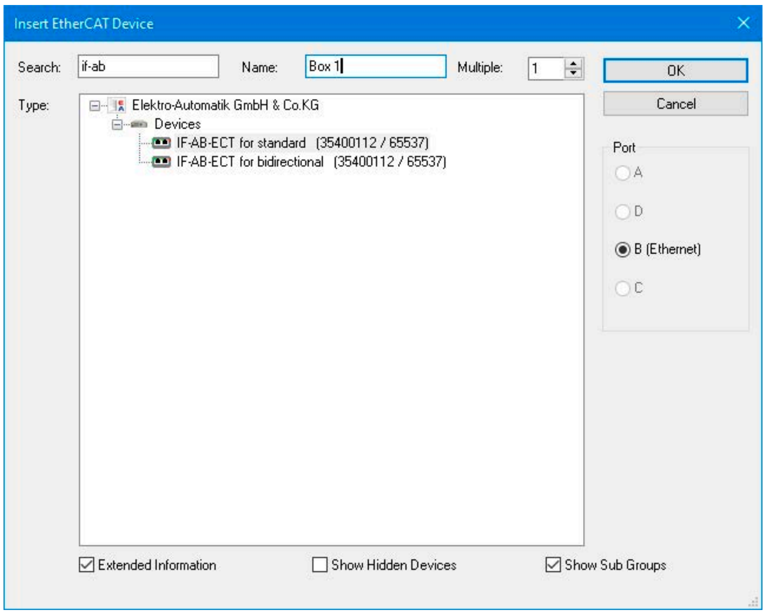
- 10000 series only: Install the EtherCAT interface module IF-AB-ECT
- Optional: install the included (USB stick) ESI (one for all 10000 series, two for 20000 series f) in the default folder of your EtherCAT master software; with TwinCAT this would be "C:\TwinCAT\<twin_cat_version>\Config\Io\EtherCAT"
- Reload the device descriptions or restart your software
- Whitelist the device descriptions, if required

9.4 Integrating your device in TwinCAT

All devices which support EtherCAT are shipped with an USB stick that holds one or several ESI files. The ESI for the 10000 series is a global one for all models in the 10000 series. It distinguishes only between bidirectional and non-bidirectional devices. The ESI, the device and the interface module don't support object mapping.

With the 20000 series, there are two ESI files delivered, one for single and one for triple channel series. They can be installed side by side and would be picked correctly during a scan. These ESIs also include all CoE objects for quicker OD load.

After installing that file and restarting the TwinCAT IDE, our EtherCAT slaves can be integrated into the setup with the **Insert EtherCAT Device** dialog and by selecting the device type:


Insert EtherCAT Device

Search: if-ab
Name: Box 1
Multiple: 1

Type:
- Elektro-Automatik GmbH & Co.KG
  - Devices
    - IF-AB-ECT for standard (35400112 / 65537)
    - IF-AB-ECT for bidirectional (35400112 / 65537)

Port: A / D / [✓ B (Ethernet)] / C

[✓] Extended Information
[ ] Show Hidden Devices
[✓] Show Sub Groups

9.5 Data objects

9.5.1 Address translation

The devices internally use the ModBus protocol and for CANopen over Ethernet communication in both directions the messages are translated. For the 10000 series, the reference for all cyclic data (PDO) and noncyclical data (SDOs) are those ModBus register lists. They are included with the device on USB stick (or are available as download) as part of the programming documentation.

With a **10000 series** device, the object dictionary of SDO objects can only uploaded from the device when accessing an online EtherCAT slave in tab "CoE" in TwinCAT. Offline objects in form of an ESI or EDS file are not available. Together with the PDO defined in the ESI file the complete list of indexes becomes accessible and allow the user to completely control the device. There is a connection between the CoE indexes and the ModBus register numbers in the lists. You can translate both back and forth.

- Translating a ModBus register to a CoE index for 10000 series devices

ModBus register number in decimal + 8193 ► convert to hexadecimal = index

Example: you want to set the device into remote control mode and want to find the corresponding CoE index. In the register list you have register number 402 for this task. Calculation: 402 + 8193 = 8595 ► converted to hexadecimal it's 0x2193, hence index 2193.

- Translating an OD index to ModBus register address for 10000 series devices

CoE index in hexadecimal - 0x2001 ► convert to decimal = register

Example: you need to know the meaning of the bits in the PDO object "Status". Find the corresponding CoE index in the index list. Here it's 21FA. Calculation: 0x21FA - 0x2001 = 0x1F9 ► converted to decimal it's 505. In the register list you will find register number 505 and the layout of the 32 bit value.

With the **20000 series**, the OD is completely represented in the ESI or can also be uploaded from the device. Furthermore, the indexes used in the OD are put into a different region starting at 0x8000. The ModBus register list is still the reference, with a few exceptions such as the "Control" object in the PDOs. This would be defined herein. The register list will also state the EtherCAT indexes for noncyclical access to every ModBus register that is supported over EtherCAT.

9.5.2 Subindexes in the RxPDO

Objects in this PDO subset are meant for device control. With date 02-27-2024 set values, limits and status can be transferred via the PDO. Everything has to be accessed using SDOs.

The ESI files define different sets of subindexes in the RxPDO of an EtherCAT slave, depending on the series:

Name	EtherCAT data type	Length in bytes	ModBus register	Short description	BT	PSB1	PSI1	PS1	PSBE1	ELR1
Status	UDINT	4	505	Device status	✓	✓	✓	✓	✓	✓
Voltage Monitor	UINT	2	507	Actual voltage on DC terminal (in %)	✓	✓	✓	✓	✓	✓
Current Monitor	UINT	2	508	Actual current on DC terminal (in %)	✓	✓	✓	✓	✓	✓
Control	UDINT	4	-	EtherCAT specific control register	✓	—	—	—	—	—
Voltage select	UINT	2	500	Set value of voltage (in %)	✓	✓	✓	✓	✓	✓
Current select	UINT	2	501	Set value of current (in %)	✓	✓	✓	✓	✓	✓
Power select	UINT	2	502	Set value of power (in %)	✓	✓	✓	✓	✓	✓
Resistance select	UINT	2	503	Set value of resistance (in %)	✓	✓	✓	✓	✓	✓
Current select sink	UINT	2	499	Set value of current (in %) for sink mode	✓	✓	—	—	✓	—
Power select sink	UINT	2	498	Set value of power (in %) for sink mode	✓	✓	—	—	✓	—
Resistance select sink	UINT	2	504	Set value of resistance (in %) for sink mode	✓	✓	—	—	✓	—
Lower limit of current (I-min)	UINT	2	9003	Adjustment limit (in %)	✓	—	—	—	—	—
Upper limit of current (I-max)	UINT	2	9002	Adjustment limit (in %)	✓	—	—	—	—	—
Lower limit of current sink (I-min)	UINT	2	9099	Adjustment limit (in %)	✓	—	—	—	—	—
Upper limit of current sink (I-max)	UINT	2	9008	Adjustment limit (in %)	✓	—	—	—	—	—
Lower limit of voltage (U-min)	UINT	2	9001	Adjustment limit (in %)	✓	—	—	—	—	—
Upper limit of voltage (U-max)	UINT	2	9000	Adjustment limit (in %)	✓	—	—	—	—	—

Name	Ether-CAT data type	Length in bytes	ModBus register	Short description	BT	PSB1	PSI1	PS1	PSBE1	ELR1
Upper limit of power (P-max)	UINT	2	9004	Adjustment limit (in %)	✓	–	–	–	–	–
Upper limit of power sink (P-max)	UINT	2	9005	Adjustment limit (in %)	✓	–	–	–	–	–
Upper limit of resistance (R-max)	UINT	2	9006	Adjustment limit (in %)	✓	–	–	–	–	–
Upper limit of resistance sink (R-max)	UINT	2	9007	Adjustment limit (in %)	✓	–	–	–	–	–

9.5.3 Subindexes in the TxPDO

Objects in this PDO subset are transferred from the slave to the master and read the device status, such as actual values.

Name	Ether-CAT data type	Length in bytes	ModBus register	Short description	BT	PSB1	PSI1	PS1	PSBE1	ELR1
Status	UDINT	4	505	Device status	✓	✓	✓	✓	✓	✓
Voltage Monitor	UINT	2	507	Actual voltage on DC terminal (in %)	✓	✓	✓	✓	✓	✓
Current Monitor	UINT	2	508	Actual current on DC terminal (in %)	✓	✓	✓	✓	✓	✓
Control	UDINT	4	-	EtherCAT specific control register	✓	–	–	–	–	–
Voltage select	UINT	2	500	Set value of voltage (in %)	✓	✓	✓	✓	✓	✓
Current select	UINT	2	501	Set value of current (in %)	✓	✓	✓	✓	✓	✓
Power select	UINT	2	502	Set value of power (in %)	✓	✓	✓	✓	✓	✓
Resistance select	UINT	2	503	Set value of resistance (in %)	✓	✓	✓	✓	✓	✓
Current select sink	UINT	2	499	Set value of current (in %) for sink mode	✓	✓	–	–	✓	–
Power select sink	UINT	2	498	Set value of power (in %) for sink mode	✓	✓	–	–	✓	–
Resistance select sink	UINT	2	504	Set value of resistance (in %) for sink mode	✓	✓	–	–	✓	–
Nominal voltage	REAL32	4	121	Rated voltage of the device or channel	✓	–	–	–	–	–
Nominal current	REAL32	4	123	Rated current of the device or channel	✓	–	–	–	–	–
Nominal power	REAL32	4	125	Rated power of the device or channel	✓	–	–	–	–	–
Max. internal resistance	REAL32	4	127	Rated resistance of the device or channel	✓	–	–	–	–	–
Min. internal resistance	REAL32	4	129	Minimum resistance of the device or channel	✓	–	–	–	–	–

9.5.3.1 PDO subindex "Control"

This object is only defined for the 20000 series. It's a 32 bit control word for various device conditions and can be extended by firmware updates. The single bits are referenced in ModBus registers and CoE indexes. With date 11/2023, following is supported:

Bit	Bit value	Short name	Description	ModBus register	CoE index
0	0x00000001	Remote on/off	0 = Remote control off; 1 = Remote control off on	402	0x8002:02
1	0x00000002	Output on/off	0 = DC terminal off; 1 = DC terminal on	405	0x8002:04
2	0x00000004	AutoOn on/off	0 = DC terminal after PF alarm -> off; 1 = DC terminal after PF alarm -> Auto	407	0x8002:05
3	0x00000008	PowerOn on/off	0 = DC terminal after leaving remote control -> off; 1 = DC terminal after leaving remote control -> Auto	408	0x8002:06
4	0x00000010	UIR on/off	0 = Resistance mode off; 1 = Resistance mode on	409	0x8002:07
5	0x00000020	Reset	1 = Reset device to factory defaults	432	0x8002:0B
6	0x00000040	ACK alarms	1 = Acknowledge and clear alarms	411	0x8002:09
7	0x00000080	MA enable on/off	0 = Master-Auxiliary (MA) deactivated; 1 = MA activated	-	0x8007:02
8	0x00000100	MA init	1 = Start MA initialization	-	0x8007:03
9	0x00000200	MA mode	0 = "Auxiliary"; 1 = "Master"	-	0x8007:01
10	0x00000400	AIM trigger	0 = AIM off; 1 = Trigger AIM once	433	0x8002:0C
11-31	0xFFFFFC00	reserved	-	-	-

9.5.4 SDO-Transfers

The noncyclical data objects for use in the EtherCAT system are defined in your device and can be downloaded from it or, in case of a 20000 series device, are implemented in the dedicated ESI. Alternatively, the list of objects can be uploaded from the device. The device needs to be online with the EtherCAT system. There is no separate documentation for the downloadable data objects. The register lists, which are part of the programming documentation, are the reference for the SDOs and explain data content and function. There is also an entire section in this document dealing with the ModBus protocol and its examples.

9.5.4.1 SDO abort codes

Following SDO transfer abort/error codes, as part of the CANopen standard, are supported by the EtherCAT slave:

Code	Description
0x06020000	Object doesn't exist in the object dictionary (ModBus register list)
0x06040043	Command not supported
0x06099911	Sub-Index doesn't exist
0x06010002	Attempt to write a read only object
0x06010002	Attempt to read a write only object
0x06070012	Too much data
0x06070013	Not enough data
0x06090030	Value range of parameter exceeded
0x08000022	Data could not be transferred or stored to the application because of the present device state
0x05040005	Out of memory
0x08000000	General error

9.6 Control of the device

Basic rule: the objects in the RxPDO can only control the most essential parameters, such as set values U, I, P and R, and with the 20000 series also DC terminal status and a few other conditions. All other functionality of the device is controlled via SDOs, in this case CoE indexes. Information about object access via SDOs with EtherCAT can be found in the documentation of the master software.

9.6.1 Use of the CoE data objects

Please refer to *"7.3. User objects"*.

The basic procedure for the access and control of a device when using EtherCAT doesn't differ from other interface types. Any device being configured and used as EtherCAT slave follows the information and instructions given in the sections *"3.2. Control locations"*, *"3.5. Special characteristics of remote control"*, *"4.3. Format of set values and resolution"*, *"4.4. Translating set values and actual values"* and *"4.6. About the register lists"*.

A. Appendix

A1. Device classes

For distinction of the different device series and especially of variants within one series, a device class number is assigned to every device. It can be read from the device (register 0 or SYSTEM:DEVICE:CLASS?) and be used to easily distinguish a power supply from an electronic load when scanning a network for units of our make.

Class	Assigned to series
16	PS 2000 B TFT Single
20	ELR 9000
21	PSI 9000 2U/3U (models from 2014)
23	PS 5000
24	PS 2000 B TFT Triple
28	PS 9000 2U/3U (models from 2014)
29	PSI 5000
30	PS 9000 1U
32	ELR 9000 TFT
33	PSI 9000 2U/3U TFT
34	ELR 9000 TFT 3W
35	PSI 9000 2U/3U TFT 3W
38	PS 9000 2U/3U 3W
39	EL 9000 B
41	ELR 5000
42	PSI 9000 DT
43	PSE 9000 3U
44	EL 9000 DT
45	PSI 9000 Slave
46	EL 9000 B Slave
47 / 50	PSI 9000 T
48 / 51	EL 9000 T
49 / 56	PS 9000 T
52	USB type B port on the simple HMI of PSI 9000 3U/WR Slave, EL 9000 B 2Q, ELR 9000 B Slave, EL 9000 B Slave
53	EL 9000 B 2Q
54	EL 9000 B 3W
55	EL 3000 B
57	PS 3000 C
58	PSB 9000
59	ELR 9000 HP
60	ELR 9000 HP 3W
61	PSB 9000 Slave
62	PSB 9000 Slave (front USB only)
63	PSB 9000 3U 3W
64	PSBE 9000
65	ELR 9000 HP Slave
66	PSB 10000
67	ELR 10000
68	PSI 10000
69	PSBE 10000
70	PSB 10000 Slave (front USB only)
81	ELR 10000 (production date since 01/2022)
82	PSI 10000 (production date since 01/2022)
83	PSB 10000 (production date since 01/2022)
84	PSBE 10000 (production date since 01/2022)
85	PS 10000
86	PU 10000
87	PUB 10000
88	PUL 10000
89	BT 20000 Triple
90	BT 20000

Legend:

TFT = Model/series with TFT touch panel (older series had LCD touch panel)

3W = Model/series with option 3W installed

1U / 2U etc. = Height of 19" enclosure in standardized height units

T = Enclosure type: Tower

DT = Enclosure type: Desktop

R = Enclosure type: wall mount

2Q = Slave unit for two-quadrants operation

EA Elektro-Automatik GmbH & Co. KG
Helmholtzstr. 31-37
41747 Viersen

Phone: +49 (2162) 3785 - 0
Fax: +49 (2162) 16230
ea1974@elektroautomatik.com

www.elektroautomatik.com

EA Elektro-Automatik Inc.
9845 Via Pasar
CA, 92126, San Diego

Phone: +1 (858) 218 2265

sales@elektroautomatik.com

www.eapowered.com