



# mPower™ DC 3xx Series

## Programmable Power Supplies

## ModBus & SCPI

For USB, GPIB, Ethernet, and AnyBus Modules

### Programming Guide and Reference

121	0x0079	x			Nominal voltage	SYSTEM:CLASS? (AUTO OFF)	float	4	2	Floating point number IEEE754
123	0x007B	x			Nominal current	SYSTEM:NOMINALVOLTAGE?	float	4	2	Floating point number IEEE754
125	0x007D	x			Nominal power	SYSTEM:NOMINALCURRENT?	float	4	2	Floating point number IEEE754
127	0x007F	x			Max. internal resistance	SYSTEM:NOMINALPOWER?	float	4	2	Floating point number IEEE754
129	0x0081	x			Min. internal resistance	SYSTEM:NOMINALPOWER?	float	4	2	Floating point number IEEE754
131	0x0083	x			Article no.	POWER:STAGE:AFTER:REMODE? (AUTO OFF)	char	40	20	ASCII
151	0x0097	x			Serial no.	POWER:STAGE:AFTER:REMODE?	char	40	20	ASCII
171	0x00AB	x	x		User text	POWER:STAGE:AFTER:REMODE?	char	40	20	ASCII
191	0x00BF	x			Firmware version (KE)	SYSTEM:CONFIG:INPUT:RESTORE? (AUTO OFF)	char	40	20	ASCII
211	0x00D3	x			Firmware version (HM)	SYSTEM:CONFIG:OUTPUT:RESTORE? (AUTO OFF)	char	40	20	ASCII
231	0x00E7	x			Firmware version (DR)	SYSTEM:CONFIG:ANALOG:PIN6 (OT PF ALL)	char	40	20	ASCII
402	0x0192	x	x		Remote mode	SYSTEM:CONFIG:ANALOG:PIN6?	enum	2	1	Coils - Remote
405	0x0195	x	x		DC output	SYSTEM:CONFIG:ANALOG:REMSB:ACTION (OFF AL)	enum	2	1	Coils - Auto-On
407	0x0197	x	x		Condition of DC output after power fail alarm	SYSTEM:CONFIG:ANALOG:REMSB:ACTION?	enum	2	1	Coils - Restan
408	0x0198	x	x		Condition of DC output after loading remote	SYSTEM:CONFIG:ANALOG:REMSB:ACTION?	enum	2	1	Coils - Power-On
409	0x0199	x	x		Operation mode (UIP UIR)	SYSTEM:CONFIG:ANALOG:REMSB:ACTION?	enum	2	1	Coils - Operation mode
410	0x019A	x			Restart of the device (warm start)	SYSTEM:CONFIG:MODE (UIP UIR)	enum	2	1	Coils - REM-SB Level
411	0x019B	x			Acknowledge alarms	SYSTEM:CONFIG:MODE?	enum	2	1	Coils - REM-SB Action
416	0x01A0	x	x		Analog interface: Reference voltage (pin VREF)	SYSTEM:CONFIG:MODE?	enum	2	1	Coils - Condition
417	0x01A1	x	x		Analog interface: REM-SB level	SYSTEM:ALARM:ACTION:OTEMPERATURE? (AUTO OFF)	enum	2	1	Coils - VREF
418	0x01A2	x			Analog interface: REM-SB action	SYSTEM:ALARM:ACTION:OTEMPERATURE?	enum	2	1	Coils - REM-SB Level
425	0x01A9	x	x		DC output/input after loading remote	SYSTEM:ALARM:ACTION:OTEMPERATURE?	enum	2	1	Coils - REM-SB Action
426	0x01AA	x	x		Function generator XY: XY mode	SYSTEM:ALARM:ACTION:OTEMPERATURE?	enum	2	1	Coils - Condition
432	0x01B0	x	x		Reset device to factory settings	SYSTEM:ALARM:ACTION:OTEMPERATURE?	enum	2	1	Coils - Condition
440	0x01B8	x	x		Analog interface: Pin 6 configuration	SYSTEM:COMMUNICATE:INTERFACE:CODE?	enum	2	1	Coils - Condition
						SYSTEM:COMMUNICATE:INTERFACE:BAUD <NR1>	enum	2	1	Coils - Condition
						SYSTEM:COMMUNICATE:INTERFACE:BAUD?	enum	2	1	Coils - Condition
441	0x01B9	x	x		Analog interface: Pin 6 configuration	SYSTEM:MS:ENABLE (ON OFF)	enum	2	1	Alarms 2
						SYSTEM:MS:LINK (MASTER SLAVE)	enum	2	1	Status-PC
442	0x01BA	x	x		Analog interface: Pin 6 configuration	SYSTEM:MS:LINK (MASTER SLAVE)	enum	2	1	Status-PC
						SYSTEM:MS:UNITS?	enum	2	1	Status-PC
500	0x01F4	x	x		Set voltage value	[SOURCE:]FUNCTION:GENERATOR:SELECT? (UI U P)	enum	2	1	0x0000 - 0xD0E5 (0 - 102%)
501	0x01F5	x	x		Set current value or impedance	[SOURCE:]FUNCTION:GENERATOR:SELECT?	enum	2	1	0x0000 - 0xD0E5 (0 - 102%)
502	0x01F6	x	x		Set power value	[SOURCE:]FUNCTION:GENERATOR:SELECT?	enum	2	1	0x0000 - 0xD0E5 (0 - 102%)
503	0x01F7	x	x		Set resistance value	[SOURCE:]FUNCTION:GENERATOR:SELECT?	enum	2	1	0x0000 - 0xD0E5 (0 - 102%)
505	0x01F9	x			Device state	[SOURCE:]FUNCTION:GENERATOR:XY:SPIN?	enum	4	2	Bit 0:4: Control location

© 2020, Marway Power Systems, Inc. All rights reserved. Some portions © Elektro-Automatik, used with permission.

mPower™, mPower DC™, mPower 3 Series™, mPower 300™, mPower 310™, and mPower 320™ are trademarks of Marway Power Systems, Inc. All other trademarks are the property of their respective owners.

## Global Support Contacts

Web: [www.marway.com](http://www.marway.com)  
Email: [support@marway.com](mailto:support@marway.com)  
[sales@marway.com](mailto:sales@marway.com)  
Phone: 800-462-7929 (7am–5pm PST)

There may be updates to this documentation and the software it describes at:  
<http://www.marway.com/mpower-dc-power-supplies>

# 1 General

1.1 Documentation Symbols .....	5
1.2 Applicability .....	5

# 2 Hardware Interfaces

2.1 USB Ports .....	6
2.1.1 Rear USB .....	6
2.1.2 Front USB .....	6
2.2 Ethernet .....	6
2.2.1 Access via HTTP .....	6
2.2.2 Access via TCP .....	7
2.3 GPIB .....	7
2.4 AnyBus Modules .....	7
2.5 AnyBus Startup Time .....	10
2.6 AnyBus Installation .....	10
2.7 AnyBus Connection Topology .....	10

# 3 Programming Introduction

3.1 Remote Communication Sources .....	11
3.2 Remote Communication Protocols .....	11
3.2.1 Protocol Detection .....	11
3.3 Remote Communication Interfaces .....	12
3.4 Control Location .....	12
3.5 Command Message Timing .....	13
3.5.1 Reading response time .....	13
3.5.2 Writing response time .....	13
3.5.3 Time between messages .....	14
3.6 Connection Timeout .....	14
3.7 Fragmented Messages .....	14
3.8 Resolution and Tolerances .....	14
3.9 Function Generator Slope .....	15
3.9.1 Externally Programmed Slope .....	16

# 4 ModBus

4.1 ModBus Overview .....	17
4.1.1 ModBus RTU .....	17
4.1.2 ModBus TCP .....	17
4.1.3 ModBus for other Interfaces .....	18
4.2 Set Value Resolution .....	18
4.2.1 Hex Percent and Decimal Value Conversion .....	18
4.3 Communication with AnyBus Modules .....	18

4.4 Communication with USB Port .....	19
4.4.1 USB driver installation .....	19
4.4.2 Discovering COM Port in Windows .....	19
4.4.3 Getting Started .....	19
4.5 Reading Register Lists .....	20
4.5.1 Function Columns .....	20
4.5.2 Data type Column .....	20
4.5.3 Access Column .....	20
4.5.4 Number of registers Column .....	20
4.5.5 Data Column .....	21
4.5.6 Profibus/Profinet slot & index Columns .....	21
4.5.7 EtherCAT SDO/PDO? Column .....	21
4.6 ModBus RTU in Detail .....	21
4.6.1 Message types .....	21
4.6.2 Slave Address .....	21
4.6.3 Functions .....	22
4.6.4 Control messages (write) .....	22
4.6.5 Query message .....	23
4.6.6 Response messages .....	24
4.6.7 The ModBus checksum .....	25
4.6.8 Communication errors .....	25
4.6.9 Examples of ModBus RTU messages .....	26
4.7 ModBus TCP in detail .....	29
4.7.1 Example for a ModBus TCP message .....	29
4.8 Specific Register Notes .....	30
4.8.1 Register 171 .....	30
4.8.2 Register 411 .....	30
4.8.3 Registers 500-503 (set values) .....	30
4.8.4 Register 505 (system status) .....	30
4.8.5 Registers 650 - 662 (master-slave configuration) .....	31
4.8.6 Registers 850 - 6695 (function generator) .....	31
4.8.7 Registers 9000 - 9009 (adjustment limits) .....	31
4.8.8 Registers 10007 - 10900 .....	31
4.8.9 Register from 12000 (advanced photovoltaics simulation, DIN EN 50530) .....	32
4.9 Remote Programming of Function Generator .....	34

# 5 SCPI protocol

5.1 Syntax .....	37
5.1.1 Concatenated commands .....	38
5.1.2 Upper and lower case .....	38
5.1.3 Long form and short form .....	38
5.1.4 Termination character .....	38
5.1.5 Errors .....	38
5.2 Value Format .....	39
5.3 Getting Started .....	40
5.3.1 Ping .....	40
5.3.2 Switch between remote and manual control .....	40

5.4	Standard IEEE commands .....	41
5.5	Status registers .....	42
5.6	Status commands .....	44
5.7	Set value commands .....	47
5.8	Measure commands .....	49
5.9	Protective feature commands .....	50
5.10	Supervision feature commands .....	51
5.11	Adjustment limit commands.....	52
5.12	Master-slave operation commands .....	53
5.13	General query commands .....	54
5.14	System configuration commands.....	55
5.14.1	General configuration commands.....	55
5.14.2	Anybus configuration commands.....	57
5.14.3	Ethernet configuration commands .....	59
5.15	Function generator commands .....	61
5.15.1	XY type: Mode selection.....	62
5.15.2	XY type: Load table data.....	62
5.15.3	XY type: Control .....	62
5.15.4	Arbitrary type: Mode and configuration.....	63
5.15.5	Arbitrary type: Load sequence data.....	63
5.15.6	Arbitrary type: Control .....	65
5.15.7	Special function: Simple PV (photovoltaics).....	66
5.15.8	Special function: FC (fuel cell) .....	67
5.16	Extended PV simulation commands .....	68
5.16.1	General configuration .....	68
5.16.2	Day trend mode configuration.....	68
5.16.3	Data recording.....	69
5.16.4	Status commands.....	70
5.16.5	Parameter commands.....	71
5.16.6	Control commands.....	72
5.16.7	Error situations.....	73
5.17	Alarm management commands.....	74
5.17.1	Reading system alarms.....	74
5.17.2	Acknowledging system alarms.....	74
5.17.3	Alarm counters.....	74
5.17.4	Example.....	74
5.18	Example applications.....	75
5.18.1	Configure and control master-slave with SCPI .....	75
5.18.2	Programming examples for PV simulation (DIN EN 50530) .....	76

## 6 Profibus & Profinet

6.1	General.....	82
6.2	Preparation .....	82
6.3	Slot configuration for Profibus .....	82
6.4	Slot configuration for Profinet.....	83
6.5	Cyclic communication via Profibus/Profinet.....	84

6.6	Acyclic communication via Profibus/Profinet .....	84
6.7	Examples for acyclic access .....	85
6.7.1	Activate/deactivate remote control.....	85
6.7.2	Send a set value.....	85
6.7.3	Read something.....	86
6.8	Data interpretation.....	87

## 7 CANopen

7.1	Preparation.....	88
7.2	User objects (indexes) .....	88
7.2.1	Translation ADI > Register.....	88
7.3	Specific examples .....	89
7.4	CANopen to ModBus differences.....	89
7.4.1	When using the arbitrary generator .....	89
7.5	Error codes .....	90

## 8 CAN

8.1	Preparation .....	91
8.2	Introduction.....	91
8.3	Message formats.....	92
8.3.1	Normal sending (writing).....	92
8.3.2	Cyclic sending (writing) .....	92
8.3.3	Querying.....	94
8.3.4	Normal reading .....	94
8.3.5	Cyclic reading .....	95
8.3.6	Message examples .....	97

## 9 EtherCAT

9.1	Preamble.....	99
9.2	Integrating your system in TwinCAT .....	99
9.3	Data objects.....	99
9.3.1	PDO object.....	100
9.3.2	SDOs .....	100
9.3.3	Use of the data objects.....	100

## Appendix A: System Classes

A.1	Class Assignments.....	101
-----	------------------------	-----

## Appendix B: 320 Series Front USB

B.1	ModBus Commands.....	102
B.2	SCPI Commands.....	102

# 1 General

## 1.1 Documentation 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) or important information for operation.



Symbol for general notices.

## 1.2 Applicability

This Programming Guide covers a multitude of hardware interfaces and software protocols available across the mPower DC Series of programmable autoranging DC power supplies. Not all features described in this Guide are available on all models of the mPower DC series.

- The mPower 300 Series models typically have USB and Ethernet remote interfaces. This Series does not have a function generator, so remote programmability for the function generator does not apply.
- The mPower 310 Series models come with either a GPIB interface, or the flexible AnyBus slot, which accepts a number of different hardware protocol modules.
- The mPower 320 Series models come with USB Type B on the rear and front panels. The port on the front panel supports a subset of the ModBus and SCPI commands relative to the rear port.

The reader will have to be aware of exactly which model and hardware interfaces are to be programmed, and therefore which software capabilities are also available. Throughout the Guide, an attempt will be made to try to make it clear which models the discussed features are applicable to. If there is no specific mention of model compatibility, it can be assumed the commands are applicable to all models.

## 2 Hardware Interfaces

### 2.1 USB Ports

#### 2.1.1 Rear USB

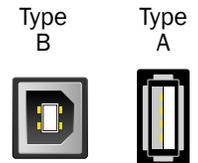
All mPower DC systems include a Type B USB port on the rear panel. This port is intended for full programmatic control of the system. Either of the ModBus RTU or SCPI protocols can be used. If you plan to use this port for remote control of the power supply, you can skip the discussions below for GPIB and AnyBus.

#### 2.1.2 Front USB

Some systems have a front USB port which might be either a Type A or a Type B connector. The Type A connector is used on the 310 Series models, whereas the Type B is used on 320 Series slave models. The Type A connector is not used for remote control purposes. Rather, it is for file access to a USB thumb drive—the details of which are in the Operating Guide.

For 320 Series models, the front Type B USB can be used for programmable access, but the functionality is limited compared to the rear port. Both ModBus RTU and SCPI can be used. Appendix B addresses the available commands.

(See also the description of the USB ports in the Operating Guide for the applicable series.)



### 2.2 Ethernet

Ethernet is not available on all models. It is not available on 320 Series models. It is optionally available on 310 Series models. It is a standard port on all 300 Series rackmount models.

When available, Ethernet can be utilized in one of two access modes: HTTP to access a basic web browser interface, or plain TCP for ModBus and SCPI commands.

#### 2.2.1 Access via HTTP

The Ethernet based modules, like for standard LAN, ModBus TCP, or Profinet, and the integrated port as featured on the 300 Series, offer a web user interface. The web UI is accessible through common web browsers by simply entering the IP address of the power supply (or the host name if your IT infrastructure has assigned it a DNS name).

In addition to presenting some configuration tools, the website gives the user full control over the system through manually typing SCPI commands. This interface primarily serves to help with command testing purposes. It's not intended to be a general purpose dashboard or control panel.

The default IP is 198.168.0.2. All network parameters for the system/network interface can be changed or reset to defaults in the setup menu of the system (locally or by remote software). As is standard for HTTP, the web interface uses port 80.

The currently active IP address, along with other network related parameters such as gateway, subnet mask, DNS address, and MAC address can also be read from an overview in the setup menu of systems where there is an on-screen setup menu (i.e. not slave units).

The CONFIGURATION page of the web interface allows for setting up network-specific parameters. This requires prior activation of remote control by command `SYST:LOCK ON`.

## 2.2.2 Access via TCP

All Anybus Ethernet modules, as well as the integrated port of 300 Series offer standard TCP access via the default port 5025 (which is user adjustable). TCP data transfer can be used for communication using SCPI or ModBus RTU (a.k.a. ModBus RTU over Ethernet). For the ModBus TCP protocol (which is not the same as ModBus over Ethernet), port 502 is used (and is not user adjustable).

The port and other network related parameters can be adjusted in the system's setup menu, from outside via USB, or through the web interface (see 2.5 above).

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



The TCP connection can be automatically disconnected by the system after a certain amount of time elapses with no data transmission. This is due to an adjustable timeout (default: 5 s). There is also another related option named **TCP keep-alive** which, if activated, makes the timeout ineffective, unless **TCP keep-alive** is not configured in the network.

## 2.3 GPIB

A GPIB interface is available as a factory installed option on some models. With this port, only SCPI can be used. ModBus RTU would still be available on the USB port.

## 2.4 AnyBus Modules

Some systems support what's called the AnyBus expansion slot. This expansion slot accepts a number of field-replaceable AnyBus Modules. These modules are described below.

Type / Name	Connectors	LED indication		Front view
CAN 2.0B / IF-AB-CAN	1x Sub-D 9pole, male	RUN	Indicates data traffic (green, flickering)	
		ERR	Will be lit (green) while a communication error is present.	
CANopen / IF-AB-CANO	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)	
RS 232 / IF-AB-RS232	1x Sub-D 9pole, male, for null modem cable	PWR	Module is powered	

Type / Name	Connectors	LED indication		Front view
Profibus / IF-AB-PBUS	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 Initialized on (green) = Initialized flashing (green) = Extended diagnosis on (red) = Exception error	
Ethernet / IF-AB-ETH1P	1x RJ45	NS	Network status: flashing (green) = default, can be ignored on (red) = Double IP, fatal error flashing (red) = Connection time-out	
Ethernet / IF-AB-ETH2P	2x RJ45	MS	Module status: flashing (green) = default, can be ignored on (red) = Exception error flashing (red) = Recoverable error	
		LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	
ProfiNET IO / IF-AB-PNET1P	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	
ProfiNET IO / IF-AB-PNET2P	2x RJ45	LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	
EtherCAT Slave / IF-AB-ECT	2x RJ45	RUN	Indicates status with flash sequences according to DR303-3 (CiA)	
		ERR	Indicates status with flash sequences according to DR303-3 (CiA)	

Type / Name	Connectors	LED indication		Front view
ModBus TCP / IF-AB-MBUS1P	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	
ModBus TCP / IF-AB-MBUS2P	2x RJ45	LINK	Connection status: on (green) = Connection established flashing (green) = Data traffic	



The CANopen module IF-AB-CANO does not feature an internal termination resistor. Thus the required bus termination resistor has to be applied by the user according to the CAN bus requirements.

---

## 2.5 AnyBus Startup Time

When integrating a unit with an AnyBus interface into an existing network or field bus, please note:

- All modules, but especially the Ethernet types which provide a website, require some startup time each time the system is powered, which may delay their network readiness. Often, an interface module is ready for communication within the time the rest of the system needs to startup, but not always. The wait time before a system is ready will have to be determined empirically based on your product and module combination.
- A readiness for operation may be indicated by a module, which has an LED for that purpose, before the needed startup time has run out. If that happens, trying to contact an Ethernet module in order to access the website may result in a time-out error reported by the browser. Just wait a little longer, and try again.

For any automated scripting, a startup ready interval should be determined through testing. This time interval should be allowed to pass before sending commands to the unit.

---

## 2.6 AnyBus Installation

The mechanical installation of the AnyBus modules is described in the mPower DC Operating Guide. This Programming Guide provides details about the specific commands which mPower DC products make use of.

Broader background information about the protocols and inter-connectivity of field buses and networks supported by the modules (ModBus, CAN, ProfiNet, etc.) can be found in publicly available documentation and similar sources. Begin by trying to locate the web sites of the organizations which manage the standards.

---

## 2.7 AnyBus Connection 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 systems in a linear topology and even to build a system level ring for extended safety against interruption. External switches can be eliminated, and the many long network cables, like when having a star-shaped topology, can be reduced to a minimum. These differences may or may not be advantages to your setup, but might be worth considering when planning for module installation.

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 cannot be considered a LAN port.

# 3 Programming Introduction

## 3.1 Remote Communication Sources

After connecting a power supply digital interface to a computer, remote software access can be gained in several ways.

- The control and monitoring software EA Power Control, available through Marway.
- LabView VIs, available through Marway.
- A custom programmed application created by the user or contractor.
- General purpose third-party software, like a terminal program that can send text messages (SCPI).
- Standardised third-party software for CAN, CANopen, Profibus, or EtherCAT etc.

## 3.2 Remote Communication Protocols

All mPower DC 3 Series power supplies support two command protocols: ModBus and SCPI.

Some form of the ModBus command protocol can be used on every communication interface available on the mPower DC 3 Series systems except for GPIB. (GPIB uses only SCPI.) SCPI can be used on most, but not all interfaces. The exact compatibility between protocols and interfaces is listed in “[3.3 Remote Communication Interfaces](#)” on page 12.

With ModBus, mPower DC uses the two standardized variations ModBus RTU and ModBus TCP, and then a subset form of the traditional ModBus RTU in conjunction with some of the other message protocols (CAN, Profibus, etc.). Each variation shares the same core register commands, but differs in its message format.

The mPower DC control system does not require that the remote protocol be declared. There is no “mode” for ModBus vs. SCPI as the remote protocol. Rather, on interfaces which are compatible with both ModBus and SCPI, commands from either protocol could technically be arbitrarily intermixed.

### 3.2.1 Protocol Detection

On many interfaces, the mPower control system uses the first byte of the remote command message to automatically distinguish the message as being either ModBus RTU or SCPI. A ModBus RTU message begins with the slave address (which is a single byte). In all mPower DC 3 Series, this slave address is 0x00. The ModBus RTU protocol uses the address of 0x00 as a broadcast address. However, for mPower DC products, they will not be on a true “bus” (i.e. RS-485). Since communication will be over USB, some form of Ethernet, or RS232, the communication is always “point-to-point” from the remote control source to the power supply.

For the built-in USB and Ethernet (not on port 502), the internal system counts on the the first byte as 0x00 to indicate the message will be a ModBus RTU. A value between 0x01 and 0x29 in the first byte will cause a ModBus communication error, whereas starting with 0x2A (ASCII character \*), the message will be considered a text message, and interpreted as a SCPI command. For AnyBus interfaces designed for specific protocols, they do not depend on this first byte distinction.



The ModBus RTU slave address is always 0. This address cannot be changed.

As a result of mPower’s auto-protocol-detection scheme, SCPI cannot be used on Ethernet TCP port 502. Since ModBus TCP requires a wrapper in which the first byte cannot be 0x00, the system cannot distinguish ModBus TCP from SCPI. Therefore, in the mPower system, port 502 (which is an IANA reserved port for ModBus TCP), is assumed to always be receiving ModBus TCP traffic. This applies also to the IF-AB-MBUS1P and IF-AB-MBUS2P Anybus modules—even though these are Ethernet ports, SCPI cannot be used.

## 3.3 Remote Communication Interfaces

As identified in the section above, while the system generally allows either ModBus or SCPI, not all interfaces support both protocols.

The table below identifies which interfaces support which protocols.

Interface	ModBus RTU	ModBus TCP	SCPI	Notes
USB (Type B)	Yes	x	Yes	
Built-in Ethernet	Yes	Yes	Yes	
Built-in GPIB	x	x	Yes	
IF-AB-CAN : CAN	PDU <sup>1</sup>	x	x	See “8. CAN”
IF-AB-CANO : CANopen	PDU <sup>1</sup>	x	x	See “7. CANopen”
IF-AB-RS232 : RS232	Yes	x	Yes	
IF-AB-PBUS : Profibus	PDU <sup>1</sup>	x	x	See “6. Profibus & Profinet”
IF-AB-PNET1P : ProfiNet	PDU <sup>1</sup>	x	x	See “6. Profibus & Profinet”
IF-AB-PNET2P : ProfiNet	PDU <sup>1</sup>	x	x	See “6. Profibus & Profinet”
IF-AB-ETH1P : Ethernet	Yes	Yes	Yes	
IF-AB-ETH2P : Ethernet	Yes	Yes	Yes	
IF-AB-MBUS1P : ModBus TCP	x	Yes	x	
IF-AB-MBUS2P : ModBus TCP	x	Yes	x	
IF-AB-ECT : EtherCAT	PDU <sup>1</sup>	x	x	See “9. EtherCAT”

<sup>1</sup> – ModBus PDUs/registers are used as the core command payload, but the communication wrappers are different.



When using ModBus on a standard Ethernet interface, use the ModBus RTU format on any available TCP port other than 502. To use ModBus TCP, port 502 must be used.

## 3.4 Control Location

Since the power supply can be controlled from the onboard controls, and from a variety of remote control interfaces, we have the concept of control location—that is, the location, or source, of where control is being managed. The power supply software concerns itself with this concept by managing whether control is allowed from the local controls only, from remote controls only, or from either location.

Models with a digital display will show the control location.

Location as displayed on the system	Description
	If there is no specific control location indicated, the system is available for remote access, and all interfaces for remote control are enabled.
Remote Analog, Remote USB, etc.	Remote control is active.
Local	Remote control is blocked, the system can only be controlled manually from the unit’s built-in controls.

All mPower DC 3 series offer a configuration setting **Allow remote control** to block remote control of the system.

Changing this setting to indicate **Local** results in remote control being blocked. Activating this block would be useful for safety purposes – a user can temporarily disable remote control in order to safely access the system to change some wiring, or just change a setting, without unexpected interference from a remote command.

Activating the **Local** condition results in the following:

- If remote control via one of the digital interfaces was currently active, remote control will be deactivated, and can be activated again later, once the **Local** condition has been deactivated again.
- If remote control via analog interface is currently active, and we assume the pin **REMOTE** has a steady signal, then remote control is interrupted only as long as condition **Local** is active. It returns automatically as soon as **Local** is deactivated. The **REMOTE** pin signal would have to be deactivated, or the plug on the analog interface be removed to keep remote inactive after the **Local** setting was deactivated.

## 3.5 Command Message Timing

Program code from a PC or PLC can send commands to a power supply faster than each command can be processed. It is the responsibility of the program code to make sure that the power supply is ready for each command. This often involves creating arbitrary delays in the program code between commands. How long those delays need to be sometimes takes experimentation. However, there are some guidelines to help establish a starting point.

### 3.5.1 Reading 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

Each method has advantages and disadvantages. The primary advantage of method 2 over method 1 is that writing and reading X times within one session is expected to be faster (shorter) compared to a full session per cycle. The primary advantage of method 1 over method 2 is that closing the port also closes the connection which makes communication more stable—especially if the time between two write-read cycles is very long.

The values in the table below have been acquired using method 1.

Series	Protocol	Typical response times		
		USB	Ethernet	CAN/CANopen
mPower 310 Series	SCPI	<15 ms	<10 ms	—
	ModBus	<10 ms	<10 ms	≈ 2 ms
mPower 300 Series	SCPI	1–3 ms	5–8 ms	—
	ModBus	≈ 1 ms	≈ 5 ms	—

### 3.5.2 Writing response time

Unlike reading, there's not a predictable response time for write commands. Each command performs unique tasks in the power supply, and some take longer to accomplish than others. Timing is further complicated by the interface used as each operates at various speeds, with varying overhead.

When developing code to automate process sequences, it will be necessary to experiment with the product, computer, and communication bus together to acquire repeatable timing data.

### 3.5.3 Time between messages

The minimum time between two messages shown below, primarily depends on the typical read time as discussed above.

Series	Minimum delay between messages	Recommended delay between messages
mPower 310 Series	USB / CAN / CANopen: 10 ms Ethernet: 15 ms	USB / CAN / CANopen: 15-20 ms Ethernet: 20 ms
mPower 300 Series	USB: 2 ms Ethernet: 8 ms	USB: 5 ms Ethernet: 15 ms

---

## 3.6 Connection Timeout

Socket connection to systems which support an Ethernet port have a connection timeout. This variable and user-adjustable timeout (see Operating Guide for the system) closes the socket connection automatically on the system side if there was no communication going on between system and 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 **TCP keep-alive** is activated, and supported in the network.

---

## 3.7 Fragmented Messages

With all communication interfaces, it is possible for the host computer (PC/PLC) to send partial messages, or messages with timing gaps between bytes. When this happens, on the power supply side, a partial command is received, there's a delay, and the rest of the command is received. Since there is no requirement for a termination byte in ModBus or SCPI, the power supply could interpret these two data burts as two commands rather than one. Of course, neither would be recognizable, and the system will respond with errors.

How long of a delay between bytes is a problem? To help create some predictability in these scenarios, a variable timeout named Com Timeout is available. This timeout defines the maximum time delay tolerated between data burts. As long as gaps between bytes is less than this timeout, the power supply will continue to accept data as a single message.

If the communication between the host and power supply experiences communication errors, and fragmented messages are suspected, the timeout can be used. Increase the value a little at a time until reliable communications are restored. 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 system can process the command.

When using SCPI, sending an additional termination character (typical LF, CR, or CRLF accepted), which is not required but accepted, will terminate the timeout immediately and let the system consider the message as completely received, so it can process.

---

## 3.8 Resolution and Tolerances

When setting output and limit values for voltage, current, power, and resistance, resolution and tolerance have to be allowed for. Resolution affects the realistic incremental steps which values can be set to. Tolerance informs the amount of inaccuracy which can be expected.

In all mPower 3 rackmount systems, there is an internal resolution of 26,214. This means that a value such as the DC output volts has 26,214 possible values (or steps) between 0 volts and the maximum nominal rating of volts for the system. A system which has a nominal rating of 80 volts has 26,214 possible steps between 0 and 80. A system which has a nominal rating of 750 volts also has 26,214 possible steps between 0 to 750. For the 80 V system, every "step" in value is an increment of  $80 \text{ V} \div 26214 \text{ steps}$ , or 0.00305 mV/step. For the 750 V system, each step is  $750 \div 26214 \text{ steps}$ , or 0.02861 mV/step.

Let's say it was desired to set the output of each system to 10.00 Vdc. For the 80 V unit,  $10.00 \text{ V} \div 0.00305 \text{ mV/step}$  results in 3,278.68 steps. A fraction of a step is not possible. So, the system is capable of either 3,278 or 3,279 steps. Multiplied back out, we end up with either 9.9979 Vdc or 10.00095 Vdc. For the 750 V unit, using the same math, the two possible results are 9.98489 Vdc or 10.0135 Vdc. Obviously these are very small differences, but differences nonetheless that you might see between setting and reading values.

Further, no control system is perfectly accurate, or perfectly repeatable. Every system has some plus or minus fluctuation of the actual result. The spec sheet for a given unit may indicate that the voltage tolerance is 0.1% (or nominal rating). This means that for an 80 Vdc system, setting any given output target can result in a value  $0.1\% \times 80 \text{ V}$ , or 80 mV. Therefore, setting a target of 24.00 Vdc means the actual measured value could be between 23.92 V and 24.08 V.

Both resolution and tolerance influence the final values, and explain why you will see small differences in set value targets and the actual measurements when reading values.

---

## 3.9 Function Generator Slope

Applicable to 310/320 Series.

With the arbitrary function generator, all sequence points have an AC part, which is used to generate sine waves, and a DC part with start and end values. When the DC start and end values are different, a slope is generated. That slope can be applied to voltage or to current.

The slope ( $\Delta U/t$  or  $\Delta I/t$ ) must meet a specific minimum value.



Minimum slope =  $0.000725 \times \text{Nominal Rated Voltage (or current)} \div \text{seconds}$

Before programming a sequence of points, the slope of those points should be calculated to make sure they meet the minimum slope. For the specific mPower unit to be used, perform the above calculation for minimum slope. Then, verify that the desired sequence of points is compatible.

Let's look at an example.

Using an mPower 310-23-050-030-001 unit (500 V and 30 A rating), a rising ramp on the current is desired. The minimum slope calculates as:  $\Delta I/t = 0.000725 \times 30 \text{ A} = 21.75 \text{ mA/s}$ .

If we wanted to ramp 0–20 A in 60 seconds, we would take  $20 \text{ amps} \div 60 \text{ seconds}$  which is 0.333 A/s or 333 mA/s. This is well above the minimum slope.

If we wanted to ramp 0–20 A in 20 minutes, we would take  $20 \text{ amps} \div (20 \times 60) \text{ seconds}$  which is 0.016 A/s or 16 mA/s. This is less than our target of 21.75 mA/s for the minimum slope. This ramp would not be possible using the function generator.

Another way to calculate this, is to determine the maximum time for a certain  $\Delta U$  or  $\Delta I$ .



Maximum ramp time =  $\text{Change in Voltage (or current)} \div \text{Minimum Slope}$

For our 20 amp slope above, we calculate  $20 \text{ A} \div 0.02175 \text{ A/s}$  which is 919.5 seconds. If we wanted a 1000 second slope duration, we would know that's too long.

What this tells us is that long ramp times over many minutes or hours are not possible using the function generator. However there is another way—which is programming a number of steps over time using external software.

### 3.9.1 Externally Programmed Slope

In the section above, we established how to determine the limits of using the function generator to create a slope in voltage or current values over time. We also established that the function generator is not suitable for slopes lasting long periods of time (many minutes to hours). However, we can create long slopes using standard external programming of output values.

Here the effective resolution comes into play (see “3.8 Resolution and Tolerances” on page 14). The unit from the example above has an effective resolution of 26214 steps which represents 0-100% = 0-30 A when working with current. For 0-20 A, it would then be 17476 steps. If we wanted to generate the 0-20 A ramp over 10 hours, we could set a new value every  $10 \text{ h} \div 17476 = \sim 2$  seconds, with the value increase every step of  $20 \text{ A} \div 17476 = \sim 1.15$  mA. This is a very small step which will be influenced by system tolerances. Therefore, it may be more reasonable to use a longer period, and a larger step. For example, an increment of 11.5 mA every 20 s, or 34 mA every minute.

Whatever the numbers used, this approach is how to create long slope periods which are beyond the limits of the internal function generator.

# 4 ModBus

## 4.1 ModBus Overview

One advantage in using the ModBus command protocol is that some form of the commands can be used on every other communication interface available on the mPower DC 3 Series systems, except for the GPIB interface. (GPIB uses only SCPI.) So, having learned ModBus, and its registers in particular, that knowledge is useful for almost all other interfaces.

Keep in mind that for the mPower products, when we talk about ModBus, there are two types: ModBus RTU and ModBus TCP. ModBus RTU is the traditional standard. ModBus TCP is a modified form.

This section of the Guide starts off with a number of topics which apply to all uses of ModBus before taking a detailed look at the traditional ModBus RTU protocol, the ModBus TCP protocol, and some register-specific notes.

A detailed listing of all registers is included in a separate PDF document *mPower-XXX-Series-Modbus-Register-List* available on our website.

### 4.1.1 ModBus RTU

The traditional ModBus RTU message consists of hexadecimal bytes in three segments called the Address, the Protocol Data Unit (PDU), and the CRC Checksum. The PDU is made up of sub-elements starting with the function code (FC), and then varies depending on the message purpose.

For Modbus RTU, all of these traditional elements are used. For example, here's the ModBus RTU message format to write a single register. Details of the message format are covered in "4.6 ModBus RTU in Detail" on page 21.

Review the table in "3.3 Remote Communication Interfaces" on page 12 to identify which interfaces that ModBus RTU can be used with.

PDU (protocol data unit)				
Address	Func. Code	Start register	Data word	CRC
0x00	0x06	0..65535	Value to write	Checksum ModBus-CRC16

For mPower DC products, the slave address (first byte) must always be 0x00. The mPower control system uses the first byte to auto-distinguish the message as being either ModBus or SCPI. A value between 0x01 and 0x29 in the first byte will cause a ModBus communication error, whereas starting with 0x2A (ASCII character \*), the message will be considered a text message, and therefore a SCPI command. (See "3.2.1 Protocol Detection" on page 11 for additional details.)

### 4.1.2 ModBus TCP

For Modbus TCP, the CRC checksum is removed, and additional message header is added. This example shows an arbitrary transaction identification of 0x4711. The rest of the ModBus TCP message is the same as ModBus RTU, except for eliminating the checksum. Information about the header can be found in "4.7 ModBus TCP in detail" on page 29.

Review the table in "3.3 Remote Communication Interfaces" on page 12 to identify which interfaces that ModBus TCP can be used with.

MBAP Header	Address	Func. Code	Start register	Data word
0x4711 0x0000 0x0006	0x00	0x06	0..65535	Value to write

### 4.1.3 ModBus for other Interfaces

While ModBus was developed for use with basic serial interfaces, the command set of the mPower ModBus protocol has been adapted to other field bus protocols. Elements of the ModBus PDU (described above), particularly the registers, are used in various ways with a distinct wrapping protocol. This still requires understanding the ModBus registers, and how data is defined and structured for ModBus messages.

The separate Registers List document, and section in this document explaining the ModBus protocol will be useful when using CAN-, Profibus-, and EtherCAT-based connectivity.

---

## 4.2 Set Value Resolution

When reading or writing set values, the numeric value is always a percent of the power supply's nominal rating (e.g. 80 V, 40A, 1000W). The percent is represented in hexadecimal where 0x0000 = 0% and 0xCCCC = 100%.

Note however, that some numeric values can be more than 100% of the nominal rating. For example, the voltage, current, and power set values can be up to 102% of the nominal rating. Therefore, for these registers, the acceptable maximum value is more than 0xCCCC (100%), and is actually 0xDOE5 (102%).

Similarly, the threshold alarms can be up to 110% of the ratings. Therefore, 0xE147 is the maximum acceptable value. When reading the actual voltage, current, or power registers can be as high as 125% of the nominal rating (0xFFFF).

All numeric values have ranges, and many have ranges which exceed 100% nominal ratings. The register reference will detail the exact range and maximum hexadecimal value.



All set values are not only limited by the unit's nominal values, but are also limited by Limits settings. Just as they would be when manually entering values, set values sent to registers which exceed the Limits settings are rejected by the power supply. It is good practice to read back the value after setting Limits to verify that the value was accepted and stored.

### 4.2.1 Hex Percent and Decimal Value Conversion

Real values have to be translated to percent values before transmitting them to the system. Also, percent values read from the system are usually translated into real values in order to process them further (in the program code managing the power supply). As described in the section above, 0xCCCC (hex) = 52,428 (decimal) = 100% nominal value (U, I, P)

Translation is done by using these formulas in your software:

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

---

## 4.3 Communication with AnyBus Modules

For all interfaces other than GPIB (which uses SCPI), RS-232 and USB (which use ModBus RTU), the core command payload is based on the ModBus registers. The field bus or network communication details create a distinct format which will be unique to the interface protocol being used (ModBus TCP, CAN, EtherCAT, Profibus, etc.). To use these interfaces

with ModBus-style commands, you will need to understand the ModBus PDU (not RTU) data structure, and then the distinct adaptations of the communication protocol you plan to use—each of which is described in a later section of this Guide (Profibus, CAN, EtherCAT, etc.).

## 4.4 Communication with USB Port

After a successful USB driver installation in the PC/PLC, and connecting to the power supply via USB cable, the system is ready for access. In Windows, the USB connection will be accessed as a COM port (which is visible in the System Manager control panel). There's no need for configuration. The driver is based upon a standardized CDC driver (Communications System Class). The typical serial settings are not effective and are ignored.

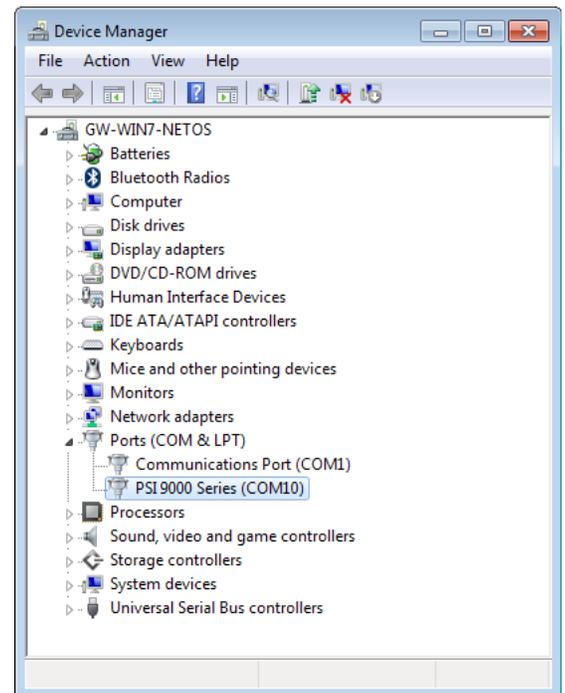
### 4.4.1 USB driver installation

The USB driver for the rear or front side USB port type B is available as a download from our website. It installs a signed driver for virtual COM ports on 32 bit or 64 bit Windows operating systems since Windows 7.

### 4.4.2 Discovering COM Port in Windows

Once the driver is installed, and one or more power supplies connected by USB to a PC, you will need to determine which COM port is used for which power supply. It is easier to do this if you attach one power supply at a time, so it is easier to find which COM port is the newest.

Attach a power supply, and turn on the AC power (it does not need any DC output connections). In Windows navigate to Control Panels > Device Manager, and look for Ports in the list, and expand it. You will find one or more entries for either PS 9000 Series or PSI 9000 Series — this is the name of the driver for the mPower DC 300 and 310/320 Series. Next to those names you will see **(COMnn)** where nn will be a single or double-digit number. That is the COM port number. As you add one unit at a time, it should be apparent which COM has been added (and therefore which unit it connects to).



### 4.4.3 Getting Started

To communicate remotely with the power supply using ModBus requires terminal software on the PC side which is able to open a COM port, and both receive and send binary messages in hexadecimal format. There are several general purpose packages which do this, and even some ModBus-aware software packages which can help with some message translation tasks. (Do an internet search for “modbus rtu terminal software” to find options.)



- In order to use remote control to change values, or switch the DC output on or off, you must activate remote control first—either with a remote command, or from the manual control panel menu.
- Remember that remote control may be blocked. See “3.4 Control Location” on page 12.

## 4.5 Reading Register Lists

Along with this programming guide, there is a register list for the 300 Series, and one for the 310/320 Series as separate PDF files. There is also a separate list specific to the 320 Series Front USB port which supports a subset of the full command set). These lists give an overview of the remote programming features available for the named series when using binary communication protocols like ModBus. They are also a reference when controlling a unit via a field bus (CAN, CANopen, Profibus, Profinet, EtherCAT) or accessing it in programming environments like LabView or MatLab.

The lists explain how the data in a binary message has to be interpreted or how a register (with CANopen or an “index” as it is called in EtherCAT) is specified. This will help the user to implement remote communication into custom software applications. Users who decide to work with SCPI command language usually do not need these lists. Later in this document, the SCPI commands are referenced in a separate chapter.

### 4.5.1 Function Columns

The heads of the 5 columns next to the ModBus address column 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 coil registers are usually writable and readable, so they’re assigned to **Read Coils (0x01)** and **Write Single Coil (0x05)**.

### 4.5.2 Data type Column

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

### 4.5.3 Access Column

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, and would not return a reasonable value when read from

RW = Register can be read or written



Writing to a register which allows write access (W, RW) is only possible during remote control.

### 4.5.4 Number of registers Column

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 the **Data length in bytes** column.

## 4.5.5 Data Column

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 and content.

## 4.5.6 Profibus/Profinet slot & index Columns

These columns (when included) are used by Profibus/Profinet users to link the registers in the register list via values **index** and **slot** to data blocks (SFBs) in the PLC software. While index is a direct parameter for the data block, the value slot has to be used to find the address of a slot, which is variable, in order to get the parameter ID. For more see “6. Profibus & Profinet”.

## 4.5.7 EtherCAT SDO/PDO? Column

This column is only available in register lists for those series which support the optionally available Anybus interface modules, and in particular the EtherCAT interface.

The column marks which of the generally per system available ModBus registers can be accessed by the CANopen over Ethernet (CoE) protocol in form of indexes. Some of the marked registers are connected to PDOs, the rest are connected as SDOs. Systems supporting the EtherCAT interface contain a downloadable data object list. Which of the registers are connected to PDOs is described in section “9. EtherCAT”.

---

## 4.6 ModBus RTU in Detail

This protocol can be used with via built-in USB interfaces, built-in Ethernet interfaces, and also with some of the optionally available AnyBus modules (see “3.3 Remote Communication Interfaces” on page 12). The addressed object when using ModBus protocol is called a 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 is not the same as ModBus TCP. ModBus TCP is run on port 502, and ModBus RTU over Ethernet can run on a port other than 502.

### 4.6.1 Message types

The ModBus message system distinguishes between control messages, query messages, and response messages. Query messages will cause the system to send a response message. Control messages only cause it to reply with an echo of the message (in order to confirm reception).

### 4.6.2 Slave Address

mPower power supplies support ModBus binary messages and the text based SCPI language with automatic protocol detection. When using ModBus, the first byte of every message (the slave address) has to be 0x00. See “3.2.1 Protocol Detection” on page 11 for more details.

## 4.6.3 Functions

The second byte of a message contains a ModBus function code (“FC”), which determines whether the message is a READ or WRITE message. It also determines, whether one or multiple registers are accessed. mPower systems support the following ModBus functions:

Function		Function name		Description	Example of use
Hex	Dec	Long	Short		
0x01	1	READ Coils	RSC, RC	Always reads 1 bit, but is returned as a full register with a 16 bit value. For example, the value 0xFF00 means logic 1, or TRUE. <u>This is different from the ModBus standard, and may cause problems with ModBus software tools.</u>	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.	Read the model name string (1-40 bytes)
0x05	5	WRITE Single Coil	WSC	Used to write 16 subsequent bits which are interpreted as 1 bit (TRUE or FALSE). <u>This is different from the ModBus standard, and may cause problems with ModBus software tools.</u>	Switch system 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. 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 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.

## 4.6.4 Control messages (write)

When sending a status, a value, multiple values, or text, the data part of the message requires to define at least the target register and one value to write. The protocol 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 fulfil the protocol requirements for the selected function code, the checksum would be expected at the position of the 7th byte. If there were additional data bytes at that position or zeros and the checksum would be at a different position in the message, the system would return an error. Therefore, the system will return an error, no matter if the message is too short or too long, because the checksum is wrong. For message examples see “4.6.9. Examples of ModBus RTU messages”.

#### WRITE Single Register

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Data word	CRC
0x00	0x06	0-65535	Value to write	Checksum ModBus-CRC16

#### WRITE Multiple Registers

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Byte 6	Bytes 7-253	Last 2 Bytes
Addr	FC	Start reg.	Number	Count	Data bytes	CRC
0x00	0x10	0-65535	0-123	Number*2	n values to write	Checksum ModBus-CRC16

#### WRITE Single Coil

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Register	Data word	CRC
0x00	0x05	0-65535	0x0000 (FALSE) or 0xFF00 (TRUE)	Checksum ModBus-CRC16



The entire 16 bits of the data word represents 1 coil only, for writing and reading.

## 4.6.5 Query message

When querying something from the system, the response is expected to be immediate, and will be of varying length, but always of the same format. 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. When querying one register with **READ Holding Registers**, the system will return two bytes, and when querying two registers it returns 4 bytes etc. For **READ Coils**, the response will always be two bytes.

For message examples see “4.6.9. Examples of ModBus RTU messages”.

#### READ Holding Registers

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Number	CRC
0x00	0x03	0-65535	Number of regs to read (1-125)	Checksum ModBus-CRC16

#### READ Coils

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Number	CRC
0x00	0x01	0-65535	Must always be 1	Checksum ModBus-CRC16



Reading coils here is not according to the ModBus specification. Reading a coil always returns 16 coils, but all 16 are considered as one bit with either TRUE (0xFF00) or FALSE (0x0000).

## 4.6.6 Response messages

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

Expected response for WRITE Single Register:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Data	CRC
0x00	0x06	0-65535	Written value echoed	Checksum ModBus-CRC16

Expected response for WRITE Single Coil:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Data	CRC
0x00	0x05	0-65535	Written value echoed	Checksum ModBus-CRC16

Expected response for WRITE Multiple Registers:

Byte 0	Byte 1	Bytes 2+3	Bytes 4+5	Bytes 6+7
Addr	FC	Start reg.	Data	CRC
0x00	0x10	0-65535	Number of written registers	Checksum ModBus-CRC16

Expected response for READ Holding Registers:

Byte 0	Byte 1	Byte 2	Bytes 3-253	Last 2 Bytes
Addr	FC	Data length in bytes	Data	CRC
0x00	0x03	2-250	Queried registers content	Checksum ModBus-CRC16

Expected response for READ Coils:

Byte 0	Byte 1	Byte 2	Bytes 3+4	Bytes 5+6
Addr	FC	Data length in bytes	Data	CRC
0x00	0x01	2	Queried bit as 1 register (always 16 coils)	Checksum ModBus-CRC16

Unexpected response (communication error):

Byte 0	Byte 1	Byte 2	Last 2 Bytes
Addr	FC		CRC
0x00	Function code + 0x80	Error code	Checksum ModBus-CRC16



A communication error can have several reasons, like a wrong checksum or when attempting to switch a system 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 "4.6.8. Communication errors".

## 4.6.7 The ModBus checksum

The checksum at the end of ModBus RTU messages is a 16 bit checksum, but is not 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](http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf) , section 2.5.1.2.

## 4.6.8 Communication errors

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

The system will return unexpected error messages in case the previously sent message is in wrong format or if the function cannot be executed by some reason. For example, when trying to write a set value with **WRITE Single Register** while the system is not in remote control. Then the message is not accepted and the system 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:

FC error	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 system:

Code		Error	Explanation
0x01	1	Wrong function code	The function code in byte 1 of the ModBus message is not supported. See "4.6.3. Functions" for supported codes. The error also occurs when trying to read or write a register with a function code for which the register is not defined.
0x02	2	Invalid address	The register address you were trying to access with read or write is not defined for your system. Every system series may have a different number of registers. Refer to the separate ModBus register list of the series your system belongs to.
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 has been transmitted in wrong byte order (high byte first instead of low byte)

Code		Error	Explanation
0x07	7	Access denied	Access to a certain register is not allowed or read only while trying to write, or vice versa. The error also occurs when trying to write to a writable address while the system is not in remote control or in remote control from a different interface
0x17	23	System in local	Indicates, that write access to the system is blocked by the <b>Local</b> condition, so only read access is possible. <b>Local</b> means that remote control is not allowed.

An example: You attempted to switch the system to remote in order to control it from PC, but instead of an echo of your message it returns something like this: 0x00 0x85 0x07 0x52 0x92. 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 0x7 which means, according to the second table above, the system has denied the access. This can have different reasons, for example that the system is already in remote control via a different interface.

## 4.6.9 Examples of ModBus RTU messages



The examples can also be used for ModBus TCP, but they need to be extended by the required ModBus TCP header and stripped of the unnecessary checksum.

### 4.6.9.1 Writing a set value



Set values are adjustable limits for the physical values Current, Voltage, Power, and Resistance (where available). They can only be written to a system 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 lists, **Set current value** is at address 501 (0x1F5) and assigned function is **WRITE Single Register**. Expecting the system to already be in remote control mode, the message to build is this:

Message to send:	Addr	FC	Start	Data	CRC	Expected response:	Addr	FC	Start	Data	CRC
	0x00	0x06	0x01F5	0x6666	0x325F		0x00	0x06	0x01F5	0x6666	0x325F

In this case, the system is expected to return an echo of your message, indicating successful execution of the command. The display of the system should now show 50% of what's the maximum current of your system. For a power supply with 510 A nominal current, it should show 255.0 A, or for a model with 170 A current rating, it should show 85 A.

### 4.6.9.2 Query all actual values at once

The system holds three readable actual values of voltage, current and power. 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. 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:	Addr	FC	Start	Data	CRC	▶
	0x00	0x03	0x01FB	0x0003	0x7417	

Possible response:	Addr	FC	Len	Data	CRC
	0x00	0x03	0x06	0x2620 0x0C9B 0x091B	0x9ECO

### 4.6.9.3 Read the nominal voltage of a system

The nominal voltage, like the other nominal values of current, power or resistance, is an important value to read from a system. They're all referenced for translating set values and actual values. It's recommended to read them from the system 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:	Addr	FC	Start	No.	CRC	▶
	0x00	0x03	0x0079	0x0002	0x1403	

Possible response:	Addr	FC	Len	Data	CRC
	0x00	0x03	0x04	0x42A00000	0xFE9

Also see 4.6.6. The response contains a float value according to IEEE754 format, which translates to 80.0.

### 4.6.9.4 Read system status

All mPower DC 3 Series systems report their system status in register 505.

Query message:	Addr	FC	Start	No.	CRC	▶
	0x00	0x03	0x01F9	0x0002	0x1417	

Possible response:	Addr	FC	Len	Data	CRC
	0x00	0x03	0x04	0x00000483	0xA992

Also see 4.6.6. The response contains the value 0x483 which states that the system is in remote control via the USB port, that the DC output is switched on, and that CC (constant current) mode is active.

#### 4.6.9.5 Switch between remote and manual control

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



The system will never switched to remote control automatically, and cannot be remote controlled in this state. Reading from all readable registers is always possible.



The system will never exit remote control automatically, unless it's switched off, or the AC supply is cut. Remote control can be terminated by a command.

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 system or read the system status), which will prevent any remote control. (See “3.4 Control Location” on page 12.)
- The system is already remotely controlled by another interface.
- The system is in setup mode, meaning a user has accessed the setup menu and not left it yet.

##### ► How to switch a system to remote control:

1. If you are using the ModBus RTU protocol, you need to create and send a message according to the description above, for example (in hex bytes): 00 05 01 92 FF 00 2D FA.
2. Once the switchover to remote control has been successful, the system will usually indicate the new condition in the display or with a LED, as well as it echoes the message as a confirmation.

In case switching to remote control would be denied by the system, because option **Allow remote control = No** is set, then the system will return an error message like 00 85 17 53 5E. According to ModBus specification, this is error 0x85 with error code 0x17.

Leaving remote control can be done in two ways: using the remote command, or by switching the system to **Local**.

##### ► How to exit remote control:

1. If you are using the ModBus RTU protocol, you need to build and send a message according to the description above, for example (in hex bytes): 00 05 01 92 00 00 6C 0A.

## 4.7 ModBus TCP in detail

This section is only about the differences to RTU. The core of a ModBus TCP message is ModBus RTU. Refer to “4.6. ModBus RTU in Detail” for more information. Differences of ModBus TCP compared to ModBus RTU:

- The message requires an additional MBAP header (6 bytes).
- The checksum is omitted (2 bytes).
- Transmission only via reserved port 502; any other port won't accept ModBus TCP frames.

The MBAP header is specified like this:

Bytes	Meaning	Explanation
0 + 1	Transaction identifier	This identifies the message. It's copied by the system in the response and is used to identify a certain message in a pool of incoming transmissions if multiple system are communicating with the PC and the response is not 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.

### 4.7.1 Example for a ModBus TCP message

The example for READ Holding Registers from “4.6.9.3. Read the nominal voltage of a system”, extended by the MBAP header (arbitrary transaction identifier 0x4711 used):

Query message:	MBAP Header	Addr	FC	Start	No.	▶
	0x4711 0x0000 0x0006	0x00	0x03	0x0079	0x0002	
Possible response:	MBAP Header	Addr	FC	Len	Data	
	0x4711 0x0000 0x0007	0x00	0x03	0x04	0x42A00000	

The example is a query for reading the system's nominal voltage. The response contains a floating point value in Data, which translates to 80(V).

---

## 4.8 Specific Register Notes

Many of the commands/registers are self-explanatory, but not all of them. Below are ones which likely need additional explanation.

### 4.8.1 Register 171

This allows to write and read a user-defined string of up to 40 characters, which is intended to be used to identify a system.

### 4.8.2 Register 411

Described for SCPI in “5.17 Alarm management commands” on page 74.

When using ModBus, this register is intended to reset alarm bits as represented in the system status (register 505, see below). Until these are reset, which is considered as an acknowledgement, the bits from previously occurred alarms remain set, even if the alarms have subsided. Alarms which are still present while register 411 is used to reset the alarm bits will of course be excluded from resetting. There is an exception: the system alarm OT (bit 19, overtemperature). This will be cleared automatically once the unit has cooled down. After resetting the alarm bits, system alarms can only be read in form of an alarm counter (registers 520 - 524).

### 4.8.3 Registers 500-503 (set values)

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

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

For 310 series units with the PV function active, the set value for current (register 501) is interpreted as irradiation value, as long as the system is in PV mode. In other words, while the PV function is running, this register does not define the current limit for your system, but a parameter called irradiation, which is commonly used in solar panel simulation. In manual operation, irradiation can be adjusted in 1% steps between 0% and 100%. With the set value of current it's also 0-100%, according to definition of register 501, but with a significantly higher resolution.

### 4.8.4 Register 505 (system status)

This register represents the system condition in one 32-bit value. Some bits are grouped, and have to be interpreted as a set. According to the register list, bits 0-4 of register 505 are a group that represents the control location (see “3.4. Control Location”). By reading this register, you can furthermore detect if the system is already in remote control to see if command “Remote mode = on” was executed by the system.

With SCPI, some but not all of these 32 bits of this register are represented in the status registers Questionable and Operation. See “5.5. Status registers”.

#### 4.8.4.1 When running master-slave

During master-slave operation (310/320 Series), the status register uses bit 29 (MSS) to indicate the master-slave safety mode, which is activated every time the master detects any problem in the communication with the slave(s), which can occur due to a connection failure or heavy electrical interferences. The master unit will then set this bit and switch off all DC outputs of the slaves being still online. Offline slaves will put themselves into a similar state and switch off DC.

After removal of the problem cause, the MS system has to be re-Initialized, which also clears the bit.

## 4.8.5 Registers 650 - 662 (master-slave configuration)

In 310/320 Series, this block of registers are used to configure the master-slave (MS) operation mode the same as you can do it from the control panel MENU. Refer to the system's Operating Guide about how the MS works and what do to in preparation of its remote control. For remote control of a MS system, it's expected to be fully wired. Before MS operation, slave units can be configured remotely, but during MS operation they can only be monitored, if required. It's, however, recommend to only control the master unit. Configuration and activation of MS operation can also be done manually and remote control can be taken over later after the master has Initialized the system.

With the MS system not being set up yet, these registers have to be used in a certain order on all units:

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

Additional steps are performed on only the master unit:

4. Initialize the MS system with register 654.
5. Optional: check with register 655, whether the initialization has been successful.
6. Optional: Query the number of initialized slaves with register 662. In case the returned number does not match the number of slave units you want to use in the MS 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 MS system to be used as value translation reference while running the MS.



The systems support reading the total ratings of voltage, current, power, and resistance even in MS mode via registers 121–129, so registers 656, 658, and 660 are actually obsolete. Alternatively they can still be used, but without the option to read the max/min resistance of the MS system. This can be done by reading the rated values from the master and dividing it by the number of units.

8. Optional: configure alarm thresholds, event thresholds and set value limits.

During MS operation, the remotely controlled master unit can be accessed like a single unit, with a few exceptions (see system manual). Set values and actual values are always percent values related to certain nominal values. Access to those registers is described in the other sections.

## 4.8.6 Registers 850 - 6695 (function generator)

While the register values fall in this zone, see “4.9. Remote Programming of Function Generator” for details.

## 4.8.7 Registers 9000 - 9009 (adjustment limits)

For SCPI, this is explained in “5.11. Adjustment limit commands”. 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).

## 4.8.8 Registers 10007 - 10900

These registers can be used to remotely configure the various built-in or optionally available digital interfaces. The registers are connected to the corresponding settings in the system's setup menu, where featured.

Contrary to manual control, the settings for the pluggable interface modules of series IF-AB (for 310 Series) can even be configured while the interface module is not yet installed.

## 4.8.9 Register from 12000 (advanced photovoltaics simulation, DIN EN 50530)

Photovoltaics simulation is a function based on the XY generator and is available on 310/320 Series power supplies.

All ModBus registers which represent parameters related to this simulation, and which can be written to the system or read from are referenced in the EN 50530 standard document. The document 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 is not different to manual handling (see user manuals of the systems) on the system's control panel or when using SCPI commands (see examples in section "5.18.2 Programming examples for PV simulation (DIN EN 50530)" on page 76). These step by step examples have an extra column in the table that holds the related ModBus register number. One of these examples (#2) converted to ModBus RTU format (percental set values translated for a system with 80 V and 170 A rating):

Configuration (before the start)

Step	Command	Description
1	00 05 01 92 FF 00 2D FA	Activate remote control
2	00 06 2E E1 00 03 91 04	Activate PV simulation mode DAYET
3	00 06 2E F0 00 00 81 00	Select technology: Manual (all required parameters must be defined, here as with commands 4-10)
4	00 10 2F 02 00 02 04 3F 4C CC CD F7 ED	Fill factor voltage (FF <sub>V</sub> ): 0,8
5	00 10 2F 04 00 02 04 3F 47 AE 14 EE FF	Fill factor current (FF <sub>I</sub> ): 0,78
6	00 10 2F 06 00 02 04 39 9D 49 52 84 57	Temperature coefficient $\alpha$ for I <sub>sc</sub> : 0,0003 / °C
7	00 10 2F 08 00 02 04 BB 44 9B A6 A1 7F	Temperature coefficient $\beta$ for U <sub>oc</sub> : -0,003 / °C
8	00 10 2F 0A 00 02 04 3D 94 7A E1 00 75	Scaling factor C <sub>U</sub> for U <sub>oc</sub> : 0,0725
9	00 10 2F 0C 00 02 04 39 66 AF CD 7F D1	Scaling factor C <sub>R</sub> for U <sub>oc</sub> : 0,00022 m <sup>2</sup> /W
10	00 10 2F 0E 00 02 04 3B 4E 70 3B A7 CE	Scaling factor C <sub>G</sub> for U <sub>oc</sub> : 0,00315 W/m <sup>2</sup>
11	00 05 2E F1 FF 00 D5 30	Select input mode: ULIK
12	00 06 2F 10 61 47 E8 A8	Set open circuit voltage: 38 V (=0x6147)
13	00 06 2F 11 08 6F 97 26	Set short-circuit current: 7 A (=0x086F)
14	00 05 2E F2 FF 00 25 30	Activate data recording
15	00 05 2E E5 00 00 D4 C4	Deactivate interpolation of day trend data
16	00 06 01 F4 61 47 A1 B7	Set global voltage limit: $\geq U_{oc}$ (=0x6147)
17	00 06 01 F6 CC CC 3C 80	Set global power limit: 100% (=0xCCCC)

Write day trend data (before the start)

Step	Command	Description
18	00 05 2E E6 FF 00 65 34	Select access mode: write
19	00 05 2E E7 FF 00 34 F4	Delete former data (should be executed every time before loading new data)

Step	Command	Description
20	00 10 2E EA 00 06 0C 00 00 00 01 44 44 66 66 00 00 03 E8 88 8A	Write 1 <sup>st</sup> day trend data set: Irradiation: 500 W/m <sup>2</sup> (=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 very first day trend data set it's expected to set 1000 ms or higher, because else the function run might fail.		
21	00 10 2E EA 00 06 0C 00 00 00 02 6D 3A 74 0D 00 00 05 DC E4 C3	Write 2 <sup>nd</sup> day trend data set: Irradiation: 800 W/m <sup>2</sup> (0x6D3A) Temperature: 28 °C (=0x740D) Dwell time: 1500 ms (=0x000005DC)
...		Write further data sets, a total of 500
519	00 10 2E EA 00 06 0C 00 00 01 F4 A3 D6 7F FF 00 00 4E 20 34 AF	Write 500. day trend data set: Irradiation: 1200 W/m <sup>2</sup> (=0xA3D6) Temperature: 35 °C (=0x7FFF) Dwell time: 20000 ms (=0x000034AF)

Control, also during simulation run

Step	Command	Description
520	00 05 2E E6 FF 00 65 34	Start simulation. The simulation will stop automatically after the time 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 system alarm.		
--	--	--

Analysis after simulation end

Step	Command	Description
521	00 03 2E F4 00 02 8D 00	Read number (n) of recorded data sets. This number is not 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	00 10 2E F6 00 02 04 00 00 00 01 6C 5C	Select first data set (index 1) for reading
523	00 03 2E F8 00 08 CD 04	Read data from data set (index) 1
...		Read further n-1 data sets:

## 4.9 Remote Programming of Function Generator

Registers 850 - 6695

The integrated function generator available with the 310/320 Series systems 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.

First of all, you need to decide which one of the two basic function generators you want to use: arbitrary or XY. Other functions, belong to the function generator, but are realised only by code.



All function generator settings and loaded data (sequences, XY table) are not stored inside the system and have to be loaded into the system every time before you can use the function generator. These data and settings are completely separate from what you can setup and define for the function generator manually when using the control panel and touch display.

### 4.9.9.1 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 have made this selection, the system cannot accept sequence point data, because the sequence data is run through a plausibility check against the system's adjustment limits.

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).

Step 4 – Set global voltage limit (register 500), if the function is applied to the current. Else set global current limit (register 501, plus 499 for PSB 9000 series), if the function is applied to voltage. Set global power limit (register 502, plus 498 for PSB 9000 series) for both modes.

Step 5 – Control the function generator with start/stop (register 850).

Step 6 – When finished, leave the function generator by deselecting your former selection of either U (register 851) or I (register 852) again.

### 4.9.9.2 Programming example for the arbitrary generator

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 set the repetition to infinite. After a 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 system 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.

The stop action and the static values are thus a little problematic for the ramp function. Why? Supposed, 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 system would also set 0 V and the voltage would drop from whatever value has been set during the function run. Conclusion: the static value of voltage 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, using only the DC part of a sequence, the parameters for the AC part (indexes 0 - 4) should be set to zero in order to avoid remainders 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:

Addr	FC	Start	Data	CRC
0x00	0x05	0x0353	0xFF00	0x7DBE

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 would not fit the width of this document's page size, the 8 float values are below each other:

Addr	FC	Start	Regs	Bytes	Data	CRC	Description
0x00	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	0x5A14	Rise time in $\mu$ s: 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:

Addr	FC	Start	Regs	Bytes	Data	CRC	Description
0x00	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	0x6AD7	Sequenz time in $\mu$ s: 180,000,000 (180 seconds = 3 minutes)

And as last step, configuration of the arbitrary generator itself:

Addr	FC	Start	Data	CRC	Description
0x00	0x06	0x035B	0x0001	0x384C	Register 859, WSR, start sequence: 1
0x00	0x06	0x035C	0x0002	0xC98C	Register 860, WSR, end sequence: 2
0x00	0x06	0x035D	0x0001	0xD84D	Register 861, WSR, sequence cycles: 1
0x00	0x06	0x01F5	0xCCCC	0xCC80	Register 501, WSR, global current limit: 100%
0x00	0x06	0x01F6	0xCCCC	0x3C80	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 systems in master-slave where those set values also limit the slaves' output.

Now the entire function setup is done and the function can be started. If the DC output of your system 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. But it's not necessary here, because the voltage shall start to rise from 0 V. In other situations where the starting level is not zero, it would be required to switch on the DC output first and wait for the voltage to settle.

For the number of sequence cycles 1 is sufficient, but it can be changed at will. The the whole function would be repeated after 3 minutes and 6 seconds. The voltage, when using a power supply, would not 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 circumvented by adding a third sequence which only uses some time for the voltage to go down.

Addr	FC	Start	Data	CRC	Description
0x00	0x05	0x0352	0xFF00	0x2C7E	Register 850, WSC, Run function

### 4.9.9.3 Procedure for the XY generator

Step 1:

Select the XY function mode with following registers:

Mode	Register
UI	854
IU	855
Simple PV (only with power supplies)	426
FC (only with power supplies)	854 (as UI mode)

Step 2:

Load the XY table data in 256 blocks of 16 values (registers 2600 - 6695). This corresponds to max. 4096 values for a measurement range of 0-125%  $U_{Nom}$  or  $I_{Nom}$ . Less data can also be loaded, for instance 3277 values for 0-100%. All values which are not set result in 0 V or 0 A.

Step 3:

This step is only required with older firmware versions. Rule of thumb: if the corresponding register list for the firmware version of your system still lists register 858, it must be used.

Submit table data (register 858).

Step 4:

Set static values which are not affected by the table

UI function: current (register 501 or CURR command) and power (register 502 or POW command)

IU function: voltage (register 500 or VOLT command) and power (register 502 or POW command)

Step 5:

Run the function generator by switching the DC input/output of your system on (register 405). For PV mode you may also want to control irradiation while the function is running. This is done by sending set values to register 501 (current), where 100% 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 you loaded in step 2.

Step 6:

Exit the function generator by deselecting your former mode setting from step 1 via the same registers.

# 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.

Be aware that not all models feature all of the commands discussed in this guide. As commands are discussed, if they are not applicable to all series, the text will indicate to which series they're compatible.

## 5.1 Syntax

The following syntax formats can occur in commands and/or responses according to "1999 SCPI Command reference" specification.

Values	This numeric value corresponds to the value in the display of the system and depends on the nominal values of the system. 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	
<NR2>	Numeric values with decimal place (floating point)	
<NR3>	Like <NR2>, but with multiplier (kilo, milli etc.)	
<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 system)	
<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: system is in remote control	
<ERR>	Error with number (-800 bis 399) 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	
<Time>	[s]s.s[s][s][s][s][s] / Default format is seconds (s.s)	
;	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 system	

## 5.1.1 Concatenated commands

It's possible to concatenate up to 5 commands in one message. Each command must be separated by a semicolon (;).

Example:

```
VOLT 80;CURR 20;POW 3kW
```

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 concatenated format, with the queried returns separated by semicolons.

## 5.1.2 Upper and lower case

SCPI uses upper case commands by convention, though the mPower control system also accepts lower case.

## 5.1.3 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 (indicating the short form), and partly in lower case (indicating the additional part of the long form).

## 5.1.4 Termination character

Some interfaces (GPIB is one) require a termination character to the message, but others don't (such as USB). When not required, the termination character is optional, but often used anyway in order to maintain software compatibility between several different interfaces.

The termination character to use is **0xA** (LF, line feed).

## 5.1.5 Errors

Errors in terms of SCPI are only communication errors. According to the standard, systems using SCPI do not return errors immediately. They have to be queried from the system. The query can occur directly with the error command (see ) or by first reading the signal bit **err** from the STB register (see "5.5. 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. The following are error strings generated by mPower systems:

Error code / error text	Description
0, "No error"	No error
-100, "Command error"	Command unknown
-102, "Syntax error"	Command syntax 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 system is in LOCAL mode
-220, "Parameter error"	Wrong parameter used
-221, "Settings conflict"	Command could not be executed because of the condition of the system (being in MENU etc.)
-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
-999, "Safety OVP"	Alarm Safety OVP (only available with specific models) has been triggered (see system manual). It requires to power cycle the system.

---

## 5.2 Value Format

In the SCPI command language real values are used in their decimal numeric text form, with or without unit identification. For example, if you wanted to set a current of 177.5 A you would use the simple command `CURR 177.5` or, with units, `CURR 177.5 A`. Likewise, values returned in responses are also in the decimal numeric text form.

## 5.3 Getting Started

### 5.3.1 Ping

It's always recommended to ping a system 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 system might send, for example (which happens to be Company Name, Part No., Serial No., HMI version, KE version, DR version, User Text (none shown)):

Protocol	Response
SCPI	Marway Power Solutions, MPW 310-12-0080-120, 1960140001, V2.18 30.08.2019 V2.28 12.08.2019 V1.6.6,

### 5.3.2 Switch between remote and manual control



The system does not switch to remote control automatically, and cannot be remotely controlled without being explicitly in remote control mode. However, *reading* status and values is always possible.



The system does not exit remote control automatically (when a remote command completes). The system must be switched off (or lose AC supply), or commanded to switch back to manual control (through control panel, or remote command).

Before you can remotely control a system, you need to switch it to remote control by sending a command. 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 system), which will prevent any remote control (see “3.4. Control Location”).
- The system is already remotely controlled by another interface
- The system in setup mode, means the user has accessed the setup menu and not left it yet

#### ► How to switch a system to remote control:

2. 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 system 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 Standard IEEE commands

To support the legacy interface standards of GPIB and IEEE 488, some of the traditional commands have been implemented.

### \*CLS

Clears the error queue and the status byte (STB).

### \*IDN?

Returns the system identification string, which contains following information, separated by commas.

1. Manufacturer
2. Model name
3. Serial number
4. Firmware version(s) (in case there are several, these are separated by a space)
5. User text (arbitrary user-definable text, as definable with SYST:CONFIG:USER:TEXT)

### \*RST

Sets the system to a defined state:

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

### \*STB?

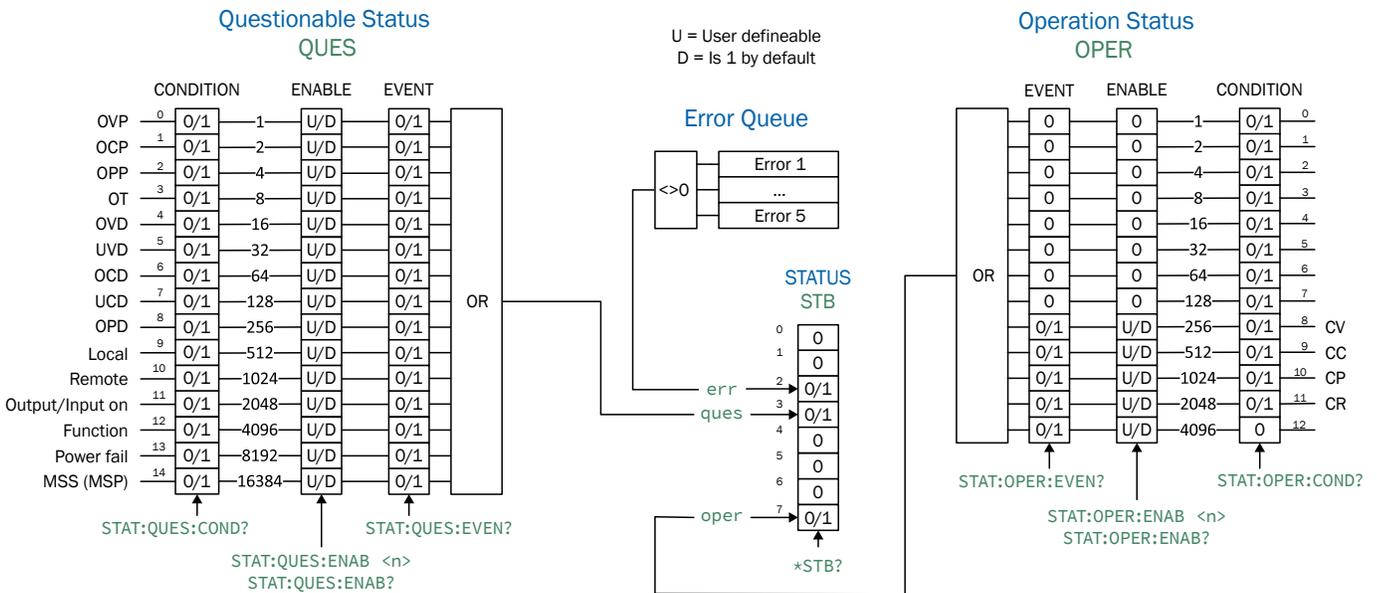
Reads the SStatus Byte register. The signal run of the various system conditions and events is illustrated in the register model below. The STB bits in particular:

- Bit 0: **sec\_ques**, Second Questionable Status Register is active (one or more events have occurred)
- Bit 1: not used
- Bit 2: **err**, Error Queue --> one or more errors 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 more events have occurred)
- Bit 4: not used
- Bit 5: not used
- Bit 6: not used
- Bit 7: **oper**, Operation Status Register is active (one or more events have occurred)

## 5.5 Status registers

Not all system conditions and alarms can be read with dedicated SCPI commands. As an alternative, the remaining system-related information are grouped in status registers. Using regular polling, the status byte (STB) can be a starting point for reading the system status. It tells which status register has recorded at least one event. Apart from that, the other status registers could also be polled directly. The difference would be that the user would have to determine which bits in the register have changed by comparing the most recent value with an older value. 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 indicates (changes to 1) that there was an event recorded in QUES or OPER register, you could read the corresponding event register of OPER and QUES, in order to find out which bits have changed in the COND register.



Events recorded in the event registers STAT:QUES:EVENT and STAT:OPER:EVENT only record positive transitions (changes from 0 to 1).



There are differences in the status registers amongst the product series based on available features. The Operating Guide of a given product series will have the details of which the features are available for that series's registers by identifying the various alarms and events.



System alarms like OVP are signaled in the subregisters CONDITION and EVENT. Each has to be acknowledged separately using commands SYST:ERR? and SYST:ERR:ALL?. These commands are considered as alarm acknowledgement, and will clear the corresponding bit in CONDITION, but only if the alarm condition is not present anymore. Alarms which have been acknowledged can be determined from the system in the form of an alarm counter. It's recommended to regularly poll alarms from the system and to query STAT:QUES? prior to SYST:ERR?.

## STATUS:QUESTIONABLE?

Reads the Questionable status CONDITION or EVENT register.

Returns a 16 bit value representing system information from the register model in 5.5.

Query Forms:

STATUS:QUESTIONABLE:CONDITION?

STATUS:QUESTIONABLE:EVENT?

STATUS:QUESTIONABLE?

Examples:

STAT:QUES? -> 3072

Reads the event register. This example reveals that bits 10 and 11 are set which identifies that **Remote = active** and **Output/input on = 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.

## STATUS:QUESTIONABLE:ENABLE <NR1>

Sets or reads the ENABLE register of the Questionable status register. The ENABLE register is a filter that enables all or single bits to signal 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.

Returns the sum of the decimal-valued bits in the register.

Query Forms:

STATUS:QUESTIONABLE:ENABLE? <0-32767>

Example:

STAT:QUES:ENAB 3081

Sets the enable register of the questionable status registers to 3072 and enables the bits **OVP** (1), **OT** (8), **Remote** (1024), and **Output/Input on** (2048) for event reporting to STB. (Note that  $1 + 8 + 1024 + 2048 = 3081$ .)

## STATUS:OPERATION?

Reads the Operation status EVENT or CONDITION register.

Returns a 16 bit value, which represents system information as defined in the register model in 5.5.

Query Forms:

STATUS:OPERATION:CONDITION?

STATUS:OPERATION:EVENT?

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 signals, that CV (constant voltage regulation) is active.

STAT:OPER:COND?

Reads the condition register of the operation status registers.

## 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 bits 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, add the bit values (see register model) and send the total to the Enable register.

Returns a value which represents the bits set to 1 in the Enable register (as seen in 5.5).

Query form:

STATUS:OPERation:ENABLE?

Value range:

<NR1> = 0, 256–3840

Since bits 0–7 are not used, the minimum value other than 0 starts with 256. Also, bit 12 is always zero, therefore the maximum sum of the bits is 3840.

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.

---

## 5.6 Status commands

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

## SYSTEM:LOCK <B0>

This command is used to activate remote control of a system. Remote control has to be activated first before you can send any command which change system status or values. Once remote control has been activated via one of the digital interfaces, only that interface is in charge.

The activation of remote control can be refused by the system 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.

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 system to switch to remote control. The system then usually indicates activated remote control either by a LED or a status text in the display.

`SYSTEM:LOCK:OWNER?`

-> `REMOTE`

Queries the lock owner regarding remote control. This can be used to verify whether the system has accepted the request to switch to remote control or not. It will return one of three different values:

`REMOTE` = System is in remote control via any of the available interfaces

`NONE` = System is not in remote control

`LOCAL` = System is in `LOCAL` condition, which denies or interrupts remote control.

## OUTPut <B0>

This command is used to switch the DC output on or off.

Query form:

`OUTPut?`

Value range:

`ON, OFF`

Examples:

`OUTP 1`

Absolute short form. Switches the DC output on if remote control is active.

`OUTPUT?`

-> `ON`

Queries the condition of the DC output, which will be returned as `ON` or `OFF`. Output state cannot be assumed as the output might be switched off due to a system alarm.

## SYSTem:ERRor?

This command is used to read a single error, or all errors, from the system's internal SCPI error queue. This queue only contains errors in relation to commands (i.e. wrong syntax, too high values etc.). It cannot return any system alarm. Those are usually queried from the system by reading bits of the status registers (see "5.5. Status registers"). You can choose either to query the next error multiple times until it says "No error," or generally query all at once. After all errors have been read from the buffer, it will be purged. (See also "4.6.8 Communication errors" on page 25.)

The queue is of type FIFO (first in, first out). This means that the first occurred error is returned first when querying them.



Querying errors with SYST:ERR? also clears bits related to system alarms in register QUESTIONable (see "5.5. Status registers"), but only if the alarm condition has already receded. The query is considered as an acknowledgement by the user. Alarms which have been acknowledged this way can then not be read from the register anymore.

### Query forms:

SYSTem:ERRor?	Queries the last or next error
SYSTem:ERRor:NEXT?	Queries the last or next error
SYSTem:ERRor:ALL?	Queries all errors in the buffer (up to 5)

### Example:

SYST:ERR?	-> 0, No error
	Absolute short form. The system replies to this query with a string that first contains an error code (see error code list) and second an error description. This is returned every time no error is present, or after all errors have been returned.
SYSTEM:ERROR:ALL?	This query will let the system return up to 5 concatenated errors in one string, separated by comma plus space.

## 5.7 Set value commands



All set values (U, I, P, R), which have dedicated single commands and which you can send to the system during remote control, are not only limited by the maximum nominal values of your particular system model, but are additionally limited by those adjustment **Limits** which you can define for manual adjustment.

### [SOURce:]VOLTage <NRf> [Unit]

Sets the output voltage limit of the system within a certain range, which is either defined by adjustment limits, or is 0–102% nominal value. When queried, this reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the voltage to the adjustable MINimum or MAXimum.

Returns a numeric value of the voltage limit setting.

Query form:

[SOURce:]VOLTage?

Value range:

<NRf> = 0–1.02 \* nominal V      102% of nominal value for the model.

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, usually 0 V.

### [SOURce:]CURRent <NRf> [Unit]

Sets the output current limit of the system within a certain range, which is either defined by adjustment limits, or is 0–102% nominal value. When queried, this reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the current to the adjustable MINimum or MAXimum.

Returns a numeric value of the current limit setting.

Query form:

[SOURce:]CURRent?

Value range:

<NRf> = 0–1.02 \* nominal V      102% of nominal value for the model.

## Example:

<code>CURR 170</code>	Absolute short form. Sets 170 A.
<code>SOUR:CURRENT 45.3A</code>	Mixed form short/long, with unit. Sets 45.3 A, unless the adjustment range of the current has been limited otherwise.
<code>SOURCE:CURRENT MAX</code>	Sets the current to the defined maximum, which is either 102% of the rated current of the system, or to the value of adjustment limit <b>I-max</b> (also see 5.11).

## [SOURce:]POWer <NRf> [Unit]

Sets the output power limit of the system within a certain range, which is either defined by adjustment limits, or is 0–102% nominal value. When queried, this reads the last setting. Alternatively, parameters MIN or MAX can be used to instantly set the current to the adjustable MINimum or MAXimum.

Returns a numeric value of the current limit setting.

### Query form:

`[SOURce:]POWer?`

### Value range:

<NRf> = 0–1.02 \* nominal V      102% of nominal value for the model.

### Examples:

<code>POW 3000</code>	Absolute short form. Sets 3000 W, unless the power adjustment range has been limited otherwise.
<code>SOUR:POWER 3.5kW</code>	Mixed form short/long, with unit and magnitude Kilo. Sets 3.5 kW (3500 W), unless the adjustment range of the power has been limited otherwise.
<code>SOURCE:POWER MIN</code>	Sets the power to the defined minimum, which is usually 0 W.

## [SOURce:]RESistance <NRf> [Unit]

Available on 310/320 Series.

This command will set the input resistance value in Ohms within a defined range, as it can be adjusted on the front panel. The mPower 310/320 units 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 system types is identical. The adjustable range can be limited with an upper adjustment limit. Alternatively, parameters MIN or MAX can be used to instantly set the resistance to the adjustable MINimum or MAXimum.

Returns a numeric value of the current limit setting.

### Query form:

`[SOURce:]RESistance?`

### Value range:

<NRf> = Min. resistance to max. resistance, according to technical specs of the model

## Examples:

<code>RES?</code>	Absolute short form. Queries the currently set resistance value.
<code>SOUR:RESISTANCE 10</code>	Mixed form short/long. Sets 10 $\Omega$ .
<code>SOURCE:RES MIN</code>	Sets the resistance to the minimum defined for the particular system model.

---

## 5.8 Measure commands

Actual values, as returned by the measuring commands, are the DC output values as they are present at the moment they are queried. They are not necessarily identical to the corresponding set values. The system constantly measures the actual values, and returns the last snapshot when queried.

The number of decimal places in the returned value will be identical to the value format in the system display and varies from model to model. (There's more about decimal places in the Operating Guides.)

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

Queries the system to return the last measured DC output voltage value in Volts.

#### Example:

`MEAS:VOLT?` → 43.50 V

Absolute short form. Queries the actual voltage. A response, which should be instant, will return a value between 0% and 125% of nominal system voltage.

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

Queries the system to return the last measured DC output current value in Amperes.

#### Example:

`MEASURE:CURRENT?` → 100.1 A

Queries the actual current only. A response, which should be instant, will return a value between 0% and 125% of nominal system current.

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

Queries the system to return the last calculated DC output power value in Watts.

#### Example:

`MEAS:POW?` → 2534 W

Absolute short form. Queries the consumed supplied power. A response, which should be instant, will return a value between 0% and 125% of nominal system power. No matter how the actual power format is in the system's display, here it will always be returned in Watts.

## MEASure:[SCALAR:]ARRay?

Queries the system to return the last measured actual values of voltage, current, and power (in that sequence), separated by commas, and with units.

Example:

MEAS:ARR? → 12.5 V, 33.3 A, 420 W

Absolute short form. A response, which should be instant, will return three values between 0% and 125% of nominal system values.

## 5.9 Protective feature commands

mPower systems feature a set of system alarms, partly for self-protection, partly for the protection of connected equipment. Additionally, there is a supervision feature which can monitor DC output attributes like voltage, current, or power for exceeding user-defined limits, and trigger user-definable actions like an acoustic alarm or shutdown of the DC output. Configuration of the supervision features can be done manually in the user profile, or by remote commands.

### [SOURCE:]VOLTage:PROTection[:LEVel] <NRf>[Unit]

This command is connected to the adjustable value OVP (overvoltage protection). The value is adjustable between 0 and 110% nominal system voltage. It defines a threshold where the system switches the DC output off whether the system has generated a voltage higher than this threshold, or the excess is coming from an outside source. When controlling a source, this feature usually serves to protect the connected load from overvoltage and thus damage. This can occur if the output voltage is accidentally adjusted to a dangerous level.

Query form:

[SOURCE:]VOLTage:PROTection[:LEVel]?

Value range:

0–1.1 \* nominal voltage of the system

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 the maximum OVP value.

### [SOURCE:]CURRent:PROTection[:LEVel] <NRf>[Unit]

This command is connected to the adjustable value OCP (overcurrent protection). The value is adjustable between 0 and 110% nominal system current. It defines a threshold where the system switches the DC output off. Once the output current reaches the threshold, the system will instantly switch the DC output off. The threshold is only effective if it's adjusted to a lower value than the output current, because otherwise the system would just limit the current, but not switch off. If current value and overcurrent protection are adjusted to the same value, the OCP has priority and will switch off rather than limit.

Query form:

[SOURCE:]CURRent:PROTection[:LEVel]?

Value range:

0–1.1 \* nominal current of the system

Example:

`CURR:PROT 100` Absolute short form. Sets the OCP threshold to 100 A.

## [SOURce:]POWer:PROTection[:LEVe]l <NRf>[Unit]

This command is connected to the adjustable value OPP (overpower protection). The value is adjustable between 0 and 110% nominal system power. It defines a threshold where the system switches the DC output off. This feature helps to protect equipment from exceeding a certain power load. Once the output power reaches the threshold, the system will instantly switch the DC output off. The threshold is only effective if it's adjusted to a lower value than the output power, otherwise the system will just limit the power, but not switch off. If power value and overpower protection are adjusted to the same value, the OPP has priority and will switch off rather than limit.

Query form:

`[SOURce:]POWer:PROTection[:LEVe]l?`

Value range:

0–1.1 \* nominal power of the system

Example:

`POW:PROT 1.5kW` Absolute short form. Sets the OPP threshold to 1.5 kW.

## 5.10 Supervision feature commands

Available on 310/320 Series.

The commands below enable the remote configuration of the supervision features (Events) of the system, related to voltage, current, or power on the DC output.

Command	Description
<code>SYSTem:CONFIg:UVD[?] &lt;NRf&gt;[Unit]</code> <code>SYSTem:CONFIg:UVD:ACTIon[?] {NONE SIGNAL WARNING ALARM}</code>	Identical to event UVD, as configurable in the system menu
<code>SYSTem:CONFIg:UCD[?] &lt;NRf&gt;[Unit]</code> <code>SYSTem:CONFIg:UCD:ACTIon[?] {NONE SIGNAL WARNING ALARM}</code>	Identical to event UCD, as configurable in the system menu
<code>SYSTem:CONFIg:OVD[?] &lt;NRf&gt;[Unit]</code> <code>SYSTem:CONFIg:OVD:ACTIon[?] {NONE SIGNAL WARNING ALARM}</code>	Identical to event OVD, as configurable in the system menu
<code>SYSTem:CONFIg:OCD[?] &lt;NRf&gt;[Unit]</code> <code>SYSTem:CONFIg:OCD:ACTIon[?] {NONE SIGNAL WARNING ALARM}</code>	Identical to event OCD as configurable in the system menu
<code>SYSTem:CONFIg:OPD[?] &lt;NRf&gt;[Unit]</code> <code>SYSTem:CONFIg:OPD:ACTIon[?] {NONE SIGNAL WARNING ALARM}</code>	Identical to event OPD, as configurable in the system menu

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

NONE = Event inactive, no supervision

SIGNAL = As soon as the event occurs, text is presented in the status field of the system display, and a bit in the Questionable Register (STAT:QUES?) is set (see “5.5. 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 system display, and a bit in the Questionable Register (STAT:QUES?) is set (see “5.5. 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 system display, as well as an acoustic alarm is initiated, the DC output is switched off, and a bit in the Questionable Register (STAT:QUES?) is set (see “5.5. Status registers”). The bit indicates, that a specific event has occurred. This can be used to record the event.



The action ALARM lets the system act similar to when a system alarm occurs. However, system alarms have priority. This means that if, for example, OVP and OVD were equal, and the output voltage reached that level, the system would initiate an OV alarm rather than an OVD event.

## 5.11 Adjustment limit commands

Adjustment limits are additional, globally effective, adjustable limits for the set values U, I, P, and R (where featured). The purpose is to narrow the standard 0–100% adjustment range and to prevent, for example, accidentally setting too high a voltage for the connected equipment. One could use overvoltage protection (OVP) for a similar purpose, but it’s generally better to prevent undesired set values in the first place.

In case a set value is sent to the system that would exceed an adjustment limit, no matter if too high or too low, the system 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** settings, as you can adjust them with the on-board controls in the setup menu of your system (except the 320 series slave units). Refer to the system manuals for details.

Command	300	310 320
[SOURce:]VOLTage:LIMit:LOW[?] <NRf>[Unit] Identical to value U-min, as configurable at the system	Y	Y
[SOURce:]VOLTage:LIMit:HIGH[?] <NRf>[Unit] Identical to value U-max, as configurable at the system	Y	Y
[SOURce:]CURRent:LIMit:LOW[?] <NRf>[Unit] Identical to value I-min, as configurable at the system	Y	Y
[SOURce:]CURRent:LIMit:HIGH[?] <NRf>[Unit] Identical to value I-max, as configurable at the system	Y	Y
[SOURce:]POWer:LIMit:HIGH[?] <NRf>[Unit] Identical to value P-max, as configurable at the system	Y	Y
[SOURce:]RESistance:LIMit:HIGH[?] <NRf>[Unit] Identical to value R-max, as configurable at the system	—	Y

## 5.12 Master-slave operation commands

The commands, as listed below, are used to remotely configure and control the master-slave mode (short: MS). This is available on the 310/320 series. (The 300 Series parallel operation using the Share bus is not the same as this Master/Slave feature.) The commands are connected to the related settings in the system setup menu. For details about MS refer to the system's operation manual.

Configuration and control require a certain procedure. Configuration should always be first, but this can be done either by remote control or manual control on the system's front panel. This allows the master-slave operation to instantly start after activating remote control.

The commands in the table below are listed in the sequence they should be used (top to bottom). The configuration of MS can be skipped if already done manually at the system's control panel or previously in remote control, and nothing has changed.

Command	Description
<code>SYSTem:MS:ENABLe {ON OFF}</code>	Enables (ON) or disables (OFF) master-slave (MS) mode
<code>SYSTem:MS:ENABLe?</code>	Queries, whether the MS is enabled or not
<code>SYSTem:MS:LINK {MASTER SLAVE}</code> <code>SYSTem:MS:LINK?</code>	Defines or queries the role of the system in the MS system: MASTER = System will be master unit SLAVE = System will be slave unit
<code>SYSTem:MS:InitializatiOn</code>	Starts the MS initialization with the given settings. Also refer to the systems'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:
<code>SYSTem:MS:CONDition?</code>	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.
<code>SYSTem:MS:UNITs?</code>	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.

## 5.13 General query commands

These commands are used to query other information from the system.

Command	300	310 320
<b>SYSTem:NOMinal:VOLTage?</b> Queries the nominal, i.e. rated input/output voltage of a single system or an Initialized master-slave system	Y	Y
<b>SYSTem:NOMinal:CURREnt?</b> Queries the nominal, i.e. rated input/output current of a single system or an Initialized master-slave system	Y	Y
<b>SYSTem:NOMinal:POWer?</b> Queries the nominal, i.e. rated input/output power of a single system or an Initialized master-slave system	Y	Y
<b>SYSTem:NOMinal:RESistance:MINimum?</b> Queries the minimum internal resistance value of a single system or an Initialized master-slave system. This value is usually not zero with electronic loads.	—	Y
<b>SYSTem:NOMinal:RESistance:MAXimum?</b> Queries the maximum internal resistance value of a single system or an Initialized master-slave system	—	Y
<b>SYSTem:System:CLAss?</b> Queries the system class and returns a value which defines to what series the system belongs to. This is an easy way to distinguish different system series.	Y	Y

## 5.14 System configuration commands

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

### 5.14.1 General configuration commands

Command	300	310 320
<b>POWer:STAGe:AFTer:REMOte { AUTO OFF }</b> <b>POWer:STAGe:AFTer:REMOte?</b> Defines, how the DC input/output of the system shall be after leaving remote control. AUTO = Last condition remains OFF = DC input/output will be switched off	Y	Y
<b>SYSTem:CONFIg:INPut:REStore[?] {AUTO OFF}</b> <b>SYSTem:CONFIg:OUTPut:REStore[?] {AUTO OFF}</b> Defines the condition of DC input/output after the system is powered. This is connected to the system setting "DC input after power on" resp. "DC output after power on". AUTO = DC input/output will be restored to the condition it had when switching the system off the last time OFF = DC input/output will always be off	Y	Y
<b>SYSTem:CONFIg:USER:TEXT &lt;SRD&gt;</b> <b>SYSTem:CONFIg:USER:TEXT?</b> Writes or queries a user-definable text of up to 40 characters permanently to the system. 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.	Y	Y
<b>SYSTem:CONFIg:ANALog:PIN6 {OT PF ALL}</b> <b>SYSTem:CONFIg:ANALog:PIN6?</b> Defines what system alarms are signaled on pin 6. OT = Pin 6 only signals OverTemperature PF = Pin 6 only signals Power Fail All = Pin 6 signals both (default)	—	Y
<b>SYSTem:CONFIg:ANALog:PIN14 {OVP OCP OPP OVP/OCP OVP/OPP OCP/OPP ALL}</b> <b>SYSTem:CONFIg:ANALog:PIN14?</b> Defines what system alarms are signaled on pin 14. There are options to signal the three system alarms OVP, OCP and OPP separately or as combination of two or signal all (logical OR)..	—	Y
<b>SYSTem:CONFIg:ANALog:PIN15 {CONT POW}</b> <b>SYSTem:CONFIg:ANALog:PIN15?</b> Defines what status is signaled on pin 15. CONT = Regulation mode CV (default) POW = DC terminal on/off	—	Y

Command	300	310 320
<p><b>SYSTem:CONFIg:ANALog:REFErence {5 10}</b>  <b>SYSTem:CONFIg:ANALog:REFErence?</b>            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 (factory setting)</p>	Y	Y
<p><b>SYSTem:CONFIg:ANALog:REMSB:LEVEl {NORMAL INVERTED}</b>  <b>SYSTem:CONFIg:ANALog:REMSB:LEVEl?</b>            Determines how pin REM-SB of the analog interface (see system manual) shall be interpreted by the system:            NORMAL = level and conditions as described in the manual (factory setting)            INVERTED = level and conditions are interpreted as inverted</p>	Y	Y
<p><b>SYSTem:CONFIg:ANALog:REMSB:ACTIon {OFF AUTO}</b>  <b>SYSTem:CONFIg:ANALog:REMSB:ACTIon?</b>            Determines the action that is caused by using pin REM-SB of the analog interface in connection with DC input/output of the system:            OFF = pin can only be used to switch the DC input/output off            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</p>	Y	Y
<p><b>SYSTem:CONFIg:MODE {UIP UIR}</b>  <b>SYSTem:CONFIg:MODE?</b>            Selects the operation mode between U/I/P and U/I/R. Both modes are available for electronic loads and also on select power supplies. By selecting U/I/R, the adjustable resistance value (command [SOURce:]RESistance resp. SINK:RESistance) is unlocked. Activated U/I/R mode can only be detected in the display from the resistance value being shown.</p>	—	Y
<p><b>SYSTem:COMMUnicate:PROTOcol:MODBus {ENABLE DISABLE}</b>  <b>SYSTem:COMMUnicate:PROTOcol:MODBus?</b>            Enables or disables ModBus protocol on the system. 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.</p>	Y	Y
<p><b>SYSTem:COMMUnicate:TIMEout {5–65535}</b>  <b>SYSTem:COMMUnicate:TIMEout?</b>            Defines a timeout in milliseconds (factory setting: 5 ms), a max. time that can elapse between two consecutive bytes, before the system considers the message as "completely received". For details refer to section 3.7.  <i>Note: this only applies for serial interfaces (USB, RS232)</i></p>	Y	Y
<p><b>SYSTem:ALARm:ACTIon:PFAiL { AUTO OFF }</b>  <b>SYSTem:ALARm:ACTIon:PFAiL?</b>            Defines, how the DC input/output of the system shall be after a power fail (PF) alarm, which could be a mains blackout or similar and after which the system could continue its work automatically.            AUTO = DC input/output condition before PF is restored            OFF = DC input/output will be switched off (default setting)</p>	Y	Y

Command	300	310 320
<b>SYSTem:ALARm:ACTion:OTEMperature { AUTO OFF }</b> <b>SYSTem:ALARm:ACTion:OTEMperature?</b> Defines, how the DC input/output of the system shall be after the system has recovered, i. e. cooled down after an overtemperature (OT) alarm. AUTO = DC input/output condition before OT is restored (default setting) OFF = DC input/output will be switched off	Y	Y

## 5.14.2 Anybus configuration commands

Available on 310 Series.

Most of the Anybus interface modules can also be remotely configured using SCPI commands, either via USB port or even via the interface itself. These settings are always saved automatically.

Command	Description		
<b>SYSTem:COMMunicate:INTerface:CODE?</b>	Returns a value, representing a model code for the installed Anybus interface module: <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 2px;">           5 = Profibus            9 = RS232            16 = CANopen            18 = ModBus TCP 1P            19 = Profinet/IO 1P            20 = Ethernet 1P         </td> <td style="width: 50%; padding: 2px;">           21 = Ethernet 2P            22 = ModBus TCP 2P            23 = Profinet/IO 2P            25 = CAN            26 = EtherCAT         </td> </tr> </table>	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
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		
<b>SYSTem:COMMunicate:INTerface:TYPE?</b>	Queries the name of the installed Anybus interface module.		
<b>SYSTem:COMMunicate:INTerface:SERial?</b>	Queries the serial number of the installed Anybus interface module.		
<b>SYSTem:COMMunicate:INTerface:ADDRess &lt;NR1&gt;</b> <b>SYSTem:COMMunicate:INTerface:ADDRess?</b>	Sets the Profibus address of the Profibus module IF-AB-PBUS or queries it. Allowed range: 0-125		
<b>SYSTem:COMMunicate:PROFibus:ID?</b>	Queries the Profibus ID of the system manufacturer.		
<b>SYSTem:COMMunicate:PROFibus:FTAG &lt;SRD&gt;</b> <b>SYSTem:COMMunicate:PROFibus:FTAG?</b>	Sets or queries the Profibus/Profinet specific “function tag”, a string of up to 32 characters		
<b>SYSTem:COMMunicate:PROFibus:LTAG &lt;SRD&gt;</b> <b>SYSTem:COMMunicate:PROFibus:LTAG?</b>	Sets or queries the Profibus/Profinet specific “location tag”, a string of up to 22 characters		
<b>SYSTem:COMMunicate:PROFibus:DATE &lt;SRD&gt;</b> <b>SYSTem:COMMunicate:PROFibus:DATE?</b>	Sets or queries the Profibus/Profinet specific “date tag”, a date/time string of up to 40 characters		
<b>SYSTem:COMMunicate:PROFibus:DESCRiption &lt;SRD&gt;</b> <b>SYSTem:COMMunicate:PROFibus:DESCRiption?</b>	Sets or queries the Profibus/Profinet specific “description” tag, a string of up to 54 characters		
<b>SYSTem:COMMunicate:PROFibus:NAME &lt;SRD&gt;</b> <b>SYSTem:COMMunicate:PROFibus:NAME?</b>	Sets or queries the Profinet specific “station name”, a string of up to 200 characters		

Command	Description																																												
<p>SYSTem:COMMunicate:INTerface:BAUD &lt;NR1&gt; SYSTem:COMMunicate:INTerface:BAUD?</p>	<p>Queries or sets the bus speed, i.e. baud rate of a CANopen or RS232 interface module. The system will only save the value. This means, with value 3 being saved and CANopen installed, it will run at 100 kbps and with RS232 installed, with 19200 Baud.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>CANopen</th> <th>CAN</th> <th>RS232</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>10 kbps</td> <td>10 kbps</td> <td>2400 Bd</td> </tr> <tr> <td>1</td> <td>20 kbps</td> <td>20 kbps</td> <td>4800 Bd</td> </tr> <tr> <td>2</td> <td>50 kbps</td> <td>50 kbps</td> <td>9600 Bd</td> </tr> <tr> <td>3</td> <td>100 kbps</td> <td>100 kbps</td> <td>19200 Bd</td> </tr> <tr> <td>4</td> <td>125 kbps</td> <td>125 kbps</td> <td>38400 Bd</td> </tr> <tr> <td>5</td> <td>250 kbps</td> <td>250 kbps</td> <td>57600 Bd</td> </tr> <tr> <td>6</td> <td>500 kbps</td> <td>500 kbps</td> <td>115200 Bd</td> </tr> <tr> <td>7</td> <td>800 kbps</td> <td>1 Mbps</td> <td>-</td> </tr> <tr> <td>8</td> <td>1 Mbps</td> <td>-</td> <td>-</td> </tr> <tr> <td>9</td> <td>Auto</td> <td>-</td> <td>-</td> </tr> </tbody> </table>	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	-	-
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	-	-																																										
<p>SYSTem:COMMunicate:CAN:BR0adcast &lt;NR1&gt; SYSTem:COMMunicate:CAN:BR0adcast?</p>	<p>Sets the CAN broadcast ID for normal CAN communication. Allowed range: 0–2047 (11 bit) resp. 0–536870911 (29 bit)</p>																																												
<p>SYSTem:COMMunicate:CAN:DLC {AUTO FILL} SYSTem:COMMunicate:CAN:DLC?</p>	<p>CAN data length setting for response messages from the system. AUTO = the number of data bytes in a CAN message from the system (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</p>																																												
<p>SYSTem:COMMunicate:CAN:FORMat {BASE EXT} SYSTem:COMMunicate:CAN:FORMat?</p>	<p>Selects the CAN address format. BASE = 11 Bit (CAN 2.0A) (default) EXT = 29 Bit (CAN 2.0B)</p>																																												
<p>SYSTem:COMMunicate:CAN:N0De &lt;NR1&gt; SYSTem:COMMunicate:CAN:N0De?</p>	<p>Sets the CAN base ID for normal CAN communication. Allowed range: 0–2047 (11 bit) resp. 0–536870911 (29 bit)</p>																																												
<p>SYSTem:COMMunicate:CAN:READ:N0De &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:N0De?</p>	<p>Sets the CAN base ID for cyclic. Also see section 8.3.5. Allowed range: 0–2047 (11 bit) resp. 0–536870911 (29 bit)</p>																																												
<p>SYSTem:COMMunicate:CAN:READ:ACTual &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:ACTual?</p>	<p>Defines the interval (in milliseconds) for the cyclic read of the system's actual values (U, I, P) over CAN interface IF-AB-CAN. Also see section 8.3.5. Allowed parameter range: 0 or 20–5000 (0 = cyclic read for this object is deactivated)</p>																																												
<p>SYSTem:COMMunicate:CAN:READ:ALIMits &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:ALIMits?</p>	<p>Defines the interval (in milliseconds) for the cyclic read of the system's adjustment limits for U and I over CAN interface IF-AB-CAN. Also see section 8.3.5. Allowed parameter range: 0 or 20–5000 (0 = cyclic read for this object is deactivated)</p>																																												

Command	Description
<pre>SYSTem:COMMunicate:CAN:READ:BLIMits &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:BLIMits?</pre>	Defines the interval (in milliseconds) for the cyclic read of the system's adjustment limits for P and R (with PSB/PSBE 9000: power and resistance of source mode) over CAN interface IF-AB-CAN. Also see section 8.3.5. Allowed parameter range: 0 or 20–5000 (0 = cyclic read for this object is deactivated)
<pre>SYSTem:COMMunicate:CAN:READ:SETS &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:SETS?</pre>	Defines the interval (in milliseconds) for the cyclic read of the system's set values (U, I, P, R) over CAN interface IF-AB-CAN. Also see section 8.3.5. Allowed parameter range: 0 or 20–5000 (0 = cyclic read for this object is deactivated)
<pre>SYSTem:COMMunicate:CAN:READ:STAT &lt;NR1&gt; SYSTem:COMMunicate:CAN:READ:STAT?</pre>	Defines the interval (in milliseconds) for the cyclic read of the system's status over CAN interface IF-AB-CAN. Also see section 8.3.5. Allowed parameter range: 0 or 20–5000 (0 = cyclic read for this object is deactivated)
<pre>SYSTem:COMMunicate:CAN:SEND:NODE &lt;NR1&gt; SYSTem:COMMunicate:CAN:SEND:NODE?</pre>	Sets the CAN base ID for cyclic send. Also see section 8.3.2. Allowed range: 0–2047 (11 bit) resp. 0–536870911 (29 bit)
<pre>SYSTem:COMMunicate:CAN:TERMination {ON OFF} SYSTem:COMMunicate:CAN:TERMination?</pre>	Switches the integrated CAN bus termination resistor ON or OFF

### 5.14.3 Ethernet configuration commands

Available on 300 and 310 Series.

The commands below are related to any Ethernet interface port, no matter if built-in or Anybus module. Some commands are only supported when an Anybus interface module is used.

Command	Description
<pre>SYSTem:COMMunicate:LAN:1SPEed[?] {AUTO 10HALF 10FULL 100HALF 100FULL}  SYSTem:COMMunicate:LAN:2SPEed[?] {AUTO 10HALF 10FULL 100HALF 100FULL}</pre>	<p>Only for IF-AB Ethernet modules (ETH, MBUS)</p> <p>Sets the communication speed of the network port(s) of Anybus Ethernet interfaces with one port (P1) or two ports (P1, P2):</p> <p>AUTO = Auto negotiation  10HALF = 10MBit, half duplex  10FULL = 10MBit, full duplex  100HALF = 100MBit, half duplex  100FULL = 100MBit, full duplex</p>
<pre>SYSTem:COMMunicate:LAN:ADDRes[?] &lt;SRD&gt;</pre>	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
<pre>SYSTem:COMMunicate:LAN:CONTRol[?] {0–65535}</pre>	Queries or sets the TCP port of the selected Ethernet interface. Default is 5025, used for ModBus RTU or SCPI communication. Systems supporting ModBus TCP have port 502 activated and reserved by default, so 502 is illegal to be set with this command.

Command	Description
<code>SYSTem:COMMunicate:LAN:DHCP[?] {ON OFF}</code>	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.
<code>SYSTem:COMMunicate:LAN:DNS1[?] &lt;SRD&gt;</code> <code>SYSTem:COMMunicate:LAN:DNS2[?] &lt;SRD&gt;</code>	Queries or sets the network address of the first DNS1 of the selected Ethernet interface, plus also for the second DNS2 (only with Anybus modules).
<code>SYSTem:COMMunicate:LAN:DOMain[?] &lt;SRD&gt;</code>	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 system in the network without knowing the IP address.
<code>SYSTem:COMMunicate:LAN:GATeway[?] &lt;SRD&gt;</code>	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 thus be left at the default setting.
<code>SYSTem:COMMunicate:LAN:HOSTname[?] &lt;SRD&gt;</code>	Queries or set the host name (refer to network terminology for details). This is a simple ASCII string of up to 54 characters.
<code>SYSTem:COMMunicate:LAN:KEEPalive[?] {ON OFF}</code>	Enables / disables the "TCP keep-alive" for the selected Ethernet interface. Also see "3.6. Connection Timeout". Default setting: OFF
<code>SYSTem:COMMunicate:LAN:MAC?</code>	Queries the MAC of the selected Ethernet interface, when physically present.
<code>SYSTem:COMMunicate:LAN:SMASK[?] &lt;SRD&gt;</code>	Queries or sets the subnet mask of the selected Ethernet interface. Format is the same as with the IP.
<code>SYSTem:COMMunicate:LAN:TIMEout[?] {0 5-65535}</code>	Defines a socket connection timeout for the selected Ethernet interface. Also see "3.6. Connection Timeout". Setting this to 0 disables the timeout. Default setting: 5 seconds
<code>SYSTem:COMMunicate:LAN:INDex {1 2}</code> <code>SYSTem:COMMunicate:LAN:INDex?</code>	Temporary switch between internal port (2) and the additional port (1, default after power-up).

## 5.15 Function generator commands

Available on 310/320 Series units.



Sequence data or table data, which can be written via SCPI commands, is not stored in the system.

The function generator is a complex part of the whole control options of the system. It can be remotely configured and controlled by a set of SCPI commands. When operating the function manager on the control panel of the system, it requires a certain procedure of setup before getting to the actual starting point. The single commands cannot enforce that procedure, so it's up to the user to use them in the correct sequence.

### 1) Select the type of generator

You need to configure the function generator at least once after the system 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% of the rated voltage or current of the system, while the arbitrary generator is used for other functions like sine wave, square wave etc.

### 2) Configure the function generator (part 1)

As a second step, the arbitrary generator requires to first select to which DC input/output value the function is assigned, voltage (U) or current (I). After that you define the number of sequences to run. This does not happen automatically when filling a certain number of sequences with data.

If you are going to use the XY generator, the next step would be to select what sort of XY curve it shall run. Depending on that selection, values you will write to the table memory are interpreted and checked for plausibility.

### 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 sequences. The number of effective sequences to run is variable, but at least 1.

The XY generator is filled with 1–4096 values.



XY data and sequence data 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 before sending the next command.

### 4) Use the function generator

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



Switching to a different function generator mode requires to first leave the function generator by sending `[SOURCE:]FUNCTION:GENERATOR:SELECT OFF`. Only after the system will allow to select another mode. Previously loaded table or sequence data will be erased when leaving function generator mode.

### 5.15.1 XY type: Mode selection

Command	Description
<pre>[SOURCE:]FUNCTION:GENERATOR:SELECT {UI IU PV NONE} [SOURCE:]FUNCTION:GENERATOR:SELECT?</pre>	Selects the run mode of XY function generator: UI = UI curve and FC curve IU = IU curve PV = Standard PV curve (for the extended EN 50530 PV curve commands refer to section 5.11) NONE = Exit function generator

### 5.15.2 XY type: Load table data

The XY generator is based on a table with up to 4096 values. Also refer to the operating manual of your unit for further details about the XY function generator.

Before actually going to fill the table with data it's required to select what kind of table is going to run, so the system can interpret and check for plausibility as soon as table data is received.

Command	Description
<pre>[SOURCE:]FUNCTION:GENERATOR:XY:LEVEL {0-4095}</pre>	1. Select one out of 4096 table entries for writing or returns the currently selected entry number.
<pre>[SOURCE:]FUNCTION:GENERATOR:XY:DATA &lt;NRF&gt; [SOURCE:]FUNCTION:GENERATOR:XY:DATA?</pre>	2. Writes a value, for example a voltage value, to the previously with :LEVEL selected table entry or returns the value.
<pre>[SOURCE:]FUNCTION:GENERATOR:XY:SUBMIT</pre>	3. Submit the data which has been written so far. Can be any number out of the max. 4096 entries. The rest would then be 0. After this the function can be started

### 5.15.3 XY type: 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 of the system on. The function will continue to run, and stops only due to system alarms, or user abort (switching DC output off).

## 5.15.4 Arbitrary type: Mode and configuration

Command	Description
[SOURce:]FUNction:GENerator:SElect {VOLTAGE CURRENT UI IU PV NONE} [SOURce:]FUNction:GENerator:SElect?	Select the type of function generator: VOLTage = Arbitrary generator for U CURRent = Arbitrary generator for I NONE = Exit function generator
[SOURce:]FUNction:GENerator:WAVE:START {1..99} [SOURce:]FUNction:GENerator:WAVE:START?	Defines the start, i.e. first sequence (1–99), or queries the last setting. If only one sequence is used, then it must be :START = :END. Sequences going to be used should also be written with data before submitting them.
[SOURce:]FUNction:GENerator:WAVE:END {1..99} [SOURce:]FUNction:GENerator:WAVE:END?	Defines the end, i.e. last sequence (1–99) or queries the last setting. If only one sequence is used, then it must be :START = :END.
[SOURce:]FUNction:GENerator:WAVE:NUMber {0..999} [SOURce:]FUNction:GENerator:WAVE:NUMber?	Defines, how often the sequence block from :START to :END is cycled through, or queries the last setting. 0 = infinite cycles 1–999 = number of cycles

## 5.15.5 Arbitrary type: Load sequence data

Sequence data should only be sent to the system 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 sequences, so one sequence is either a complete function or just a part of it. When started, the function generator will execute the sequences from start sequence to end sequence, as defined by the user. With every sequence being variable, the resulting function can be quite complex. The sequence data is loaded into the system with three commands and in a specific order like this:

Command	Description
[SOURce:]FUNction:GENerator:WAVE:LEVel {1–99} [SOURce:]FUNction:GENerator:WAVE:LEVel?	1. Selects a sequence (similar to HMI access) to write, or queries the currently selected sequence number
[SOURce:]FUNction:GENerator:WAVE:INdEx {0–7} [SOURce:]FUNction:GENerator:WAVE:INdEx?	2. For the selected sequence, a set of parameters can be configured. This command selects the parameter between 0 and 7 with value INdEx. The next command (:DATA), is then used to write a value. The indexes are explained below. Can also be used to query the current index.
[SOURce:]FUNction:GENerator:WAVE:DATA <NRf> [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. Can also be used to query the last value.
[SOURce:]FUNction:GENerator:WAVE:SUBmit	4. Submits all data. Without sending this command, the FG can be started, but will run with all values being zero



The AC and DC start and end values have a dependency on each other. Rule of thumb: use of the AC part requires to set DC values first, otherwise the AC values are not accepted by the system and an error is put into the SCPI error buffer. The DC values (start, end) must not be smaller than the related AC values (start, end).

It can be useful to read back a value that was just written to the system, in order to verify whether it has been accepted or not. Alternatively, you may read the error queue.

When adjusting parameters for the arbitrary function generator manually on the system's control panel, they are limited to each so the resulting signal will work as expected. But here, in remote control, there will be no plausibility check and so it's up to the user to write correct values.

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, for instance, 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 will 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.

In relation to the adjustments for a function as they can be done on the system's front panel, following indexes are selectable and readable/writable with sub command :DATA.

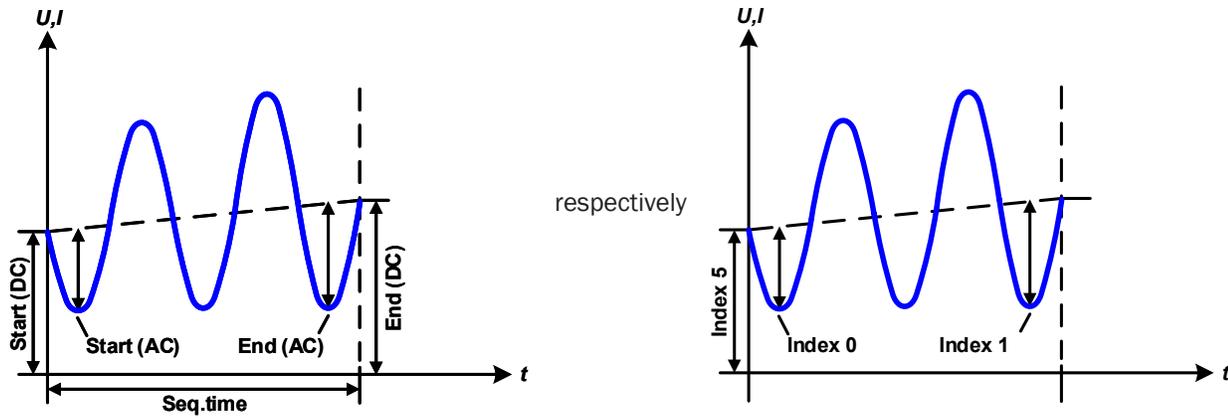
Index	Parameter	Data type	Unit	Value range	Note
0	Start value (amplitude)	Float	A, V	0–Nominal value (U or I)	For AC part only
1	End value (amplitude)	Float	A, V	0–Nominal value (U or I)	For AC part only
2	Start frequency in Hz	Integer	Hz	0–10000	For AC part only
3	End frequency in Hz	Integer	Hz	0–10000	For AC part only
4	Start angle in °	Integer	–	0–359	For AC part only
5	Start level (offset)	Float	A, V	0–Nominal value (U or I)	For AC and DC part
6	End level (offset)	Float	A, V	0–Nominal value (U or I)	For AC and DC part
7	Sequence time	Float	s	0.0001–36000	



In case start and end value (indexes 0+1 and indexes 5+6) are not equal, the system expects a certain minimum change of  $\pm 0.058\%/s$  of  $\pm 9.3$  Hz/s for the start and end frequency (indexes 2+3) over the sequence time. Therefore, it is not possible to let the input current rise by 1 A over 1 h, because this exceeds the internal set value resolution.

Another example: with the sequence time being set to 2 s, a start frequency of 1 Hz and end frequency of 10 Hz would not be accepted, because the difference is only 9 Hz/s, but start frequency of 30 Hz and end frequency of 5 Hz would.

Parameter assignment illustrated by an example curve:



### 5.15.6 Arbitrary type: Control

Contrary to the XY generator where the curve is immediately active when switching on the DC output, the arbitrary generator requires run control by command (below). The run cannot be paused. This means, once the function is stopped, no matter by what reason, the next start will run from the beginning, the first sequence in use.

Command	Description
<code>[SOURCE:]FUNCTION:GENERATOR:WAVE:STATE {RUN STOP}</code>	Starts/stops the arbitrary function generator
<code>[SOURCE:]FUNCTION:GENERATOR:WAVE:STATE?</code>	or queries the STATE

## 5.15.7 Special function: Simple PV (photovoltaics)



For the extended PV function EN 50530 see “5.16. Extended PV simulation commands”.

The photovoltaics function (PV), available in the 310/320 Series, is based on the XY function generator. When programming a PV using the control panel of the 310 Series, you only have to set up 4 parameters, and the system calculates the table. However, in remote control mode, it's necessary to load a complete, precalculated table with 4096 values into the system (using scripted SCPI commands).

There are a few options to get the table data calculated and entered:

1. Use the control panel, enter four PV simulation parameters, then let the system calculate and load the table.
2. Use external tools to create a CSV file. Load that file using a thumb drive into the system from the control panel.
3. Create a custom software script using SCPI and directly enter the PV table using remote control.

Regardless of which method is used, once a table is loaded in the system, the control panel can be used to save the table to a thumb drive in the front USB port. (If you're using custom remote control scripting, saving the table to the thumb drive isn't necessary, as you would just reload the table by remote control. Still, it might be useful for archiving and using manually later.)

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

### 5.15.7.1 Loading data for the simple PV function

No.	Command	Description
1	<code>FUNC:GEN:SEL PV</code>	Select the PV function for the XY generator. By sending this command the system will switch to function generator mode
2	<code>FUNC:GEN:XY:LEVEL 0</code>	Subsequently write all XY table values for the PV curve into the system
3	<code>FUNC:GEN:XY:DATA 150</code>	
...		
8192	<code>FUNC:GEN:XY:LEVEL 4095</code>	
8193	<code>FUNC:GEN:XY:DATA 0</code>	
8194	<code>FUNC:GEN:XY:SUB</code>	Submit the written data

### 5.15.7.2 Irradiation

During function run, the irradiation value which simulates different light situations, can be adjusted. The irradiation impacts the short-circuit current ( $I_{sc}$ ) as a factor.

Command	Description
<code>[SOURce:]IRRadiation &lt;NR1&gt;</code> <code>[SOURce:]IRRadiation?</code>	Adjust or queries the irradiation value during the solar panel simulation in a range of {0–100} percent, which affects the DC current and shifts the MPP on the Y axis

### 5.15.7.3 Setting limits

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 does not interfere the simulation. The voltage here should be set to the open circuit voltage (Uoc), alternatively to maximum:

No.	Command	Description
8195	VOLT MAX	Set voltage to max, independent from the model
8196	POW MAX	Set power to max, independent from the model

After all 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 vertically on the Y axis. For an illustration of the PV curve, refer to your unit's manual.

### 5.15.7.4 Controlling the system during the PV function run

No.	Command	Description
8197	OUTP ON	Switch the DC output of your system on to make the function start
8198	IRR 85	Set irradiation to 85% (example) or any other value between 0 and 100
8199	OUTP OFF	Switch the DC output of your system off to make the function stop
8200	FUNC:GEN:SEL NONE	Parameter NONE selects no function generator type and leaves the function generator mode

### 5.15.8 Special function: FC (fuel cell)

The fuel cell function (FC) is based on the XY function generator, and is available in the 310 and 320 Series. In remote control mode, it's only possible to load a complete, precalculated table with 4096 values into the system. A fuel cell curve is essentially a voltage-current (UI) curve—so, using the control panel you would select XY Table, then UI Table. Using SCPI command you would select UI mode. See “5.15.1. XY type: Mode selection” for the SCPI commands.

When programming a FC function using the control panel of the 310 Series, you only have to set up 4 parameters, and the system calculates the table. However, in remote control mode, it's necessary to load a complete, precalculated table with 4096 values into the system (using scripted SCPI commands).

There are a few options to get the table data calculated and entered:

1. Use the control panel, enter four PV simulation parameters, then let the system calculate and load the table.
2. Use external tools to create a CSV file. Load that file using a thumb drive into the system from the control panel.
3. Create a custom software script using SCPI and directly enter the PV table using remote control.

Regardless of which method is used, once a table is loaded in the system, the control panel can be used to save the table to a thumb drive in the front USB port. (If you're using custom remote control scripting, saving the table to the thumb drive isn't necessary, as you would just reload the table by remote control. Still, it might be useful for archiving and using manually later.)

Use the same command set as listed for “5.15.7.1. Loading data for the simple PV function”

## 5.16 Extended PV simulation commands

Available in the 310/320 Series.

### 5.16.1 General configuration

Command	Description
<pre> FUNCTION:PHOTOvoltaics:MODE {OFF ET UI DAYET DAYUI} FUNCTION:PHOTOvoltaics:MODE? </pre>	<p>General mode selection for the PV simulation</p> <p>OFF = simulation mode off (default)</p> <p>ET = Continuous mode, temperature and irradiation can be varied during simulation</p> <p>UI = Continuous mode, voltage and current of the MPP can be varied during simulation</p> <p>DAYET = Day trend mode, no values can be varied, the data set consists of an index, a temperature value, an irradiation value and a dwell time</p> <p>DAYUI = Day trend mode, no values can be varied, the data set consists of an index, MPP voltage and current values and a dwell time</p>
<pre> FUNCTION:PHOTOvoltaics:IMODE {MPP ULIK} FUNCTION:PHOTOvoltaics:IMODE? </pre>	<p>Input mode (applies to all modes selectable with :MODE command, also see matrix and examples in <a href="#">5.18.2</a>)</p> <p>MPP = The base values to calculate the PV curve from are entered as <math>U_{mpp}</math> and <math>I_{mpp}</math>. These values are adjustable simulation mode UI (default)</p> <p>ULIK = The base values to calculate the PV curve from are entered as <math>U_{oc}</math> (open circuit voltage) and <math>I_{sc}</math> (short-circuit current). These values are adjustable in simulation mode ET</p>

### 5.16.2 Day trend mode configuration

The below listed commands can only be used if any of the day trend modes DAYET or DAYUI (see above) has been set before and will else 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
<pre> FUNCTION:PHOTOvoltaics:DAY:INTERpolate {ON OFF} FUNCTION:PHOTOvoltaics:DAY:INTERpolate? </pre>	<p>Only for modes DAYET and DAYUI:</p> <p>ON = Interpolation on</p> <p>OFF = Interpolation off (default)</p>
<pre> FUNCTION:PHOTOvoltaics:DAY:MODE {READ WRITE} FUNCTION:PHOTOvoltaics:DAY:MODE? </pre>	<p>Only for modes DAYET and DAYUI:</p> <p>Day trend mode data access type</p> <p>READ = Read only (default)</p> <p>WRITE = Write only</p>
<pre> FUNCTION:PHOTOvoltaics:DAY {CLEAR} </pre>	<p>Only for modes DAYET and DAYUI:</p> <p>Clear all data</p>

Commands	Description
FUNCtion:PHOTovoltaics:DAY:INDEX {1-100000} FUNCtion:PHOTovoltaics:DAY:INDEX?	Only for modes DAYET and DAYUI: Select the data index before re-reading day trend data. For writing day trend data, this index values is ignored. Use the index in FUNC:PHOT:DAY:DATA command instead.
FUNCtion:PHOTovoltaics:DAY:DATA {<NR1>, <NRf>, <NRf>, <NR1>} FUNCtion:PHOTovoltaics:DAY:DATA?	Only for modes DAYET and DAYUI: Write one set of day trend data (4 values) or read them back from it, which requires prior index selection. Depending on the selected day trend mode, different data is returned upon read or must be provided when writing.  Mode DAYET: 1. value = index, range: 1- 100000 2. value = irradiation in W/m <sup>2</sup> , range: 0-1500 3. value = temperature in °C, range: -40...+80 4. value = Dwell time of index in ms, range: 500-1800000 (^=0,5s...0,5h)  Mode DAYUI: 1. value = index, range: 1-100000 2. value = Umpp in V, range: 0...rated voltage 3. value = Impp in A, range: 0...rated current 4. value = Dwell time of index in ms, range: 500-1800000 (^=0,5s...0,5h)
FUNCtion:PHOTovoltaics:TECHnology {MAN CSI THIN} FUNCtion:PHOTovoltaics:TECHnology?	Panel technology preselection. Determines if some simulation parameters are fixed or accessible MAN = Manual mode (all parameters unlocked) CSI = cSi technology panel (default) THIN = thin film technology panel

### 5.16.3 Data recording

The system 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 set by set. In case they are need to be saved, they should be read as long as the unit is powered, because the internal memory data is not stored.

Command	Description
FUNCtion:PHOTovoltaics:RECORD:ACTIVE {ENABLE DISABLE} FUNCtion:PHOTovoltaics:RECORD:ACTIVE?	Data recording ENABLE = activated DISABLE = deactivated (default)
FUNCtion:PHOTovoltaics:RECORD {CLEAR}	Clear recorded data

Command	Description
<code>FUNCTION:PHOTOvoltaics:RECORD:NUMBER?</code>	Number of already recorded data sets (1-576000)
<code>FUNCTION:PHOTOvoltaics:RECORD:INDEX {1-576000}</code> <code>FUNCTION:PHOTOvoltaics:RECORD:INDEX?</code>	Set or read the index number prior to read a data set with <code>:DATA?</code> command
<code>FUNCTION:PHOTOvoltaics:RECORD:DATA?</code>	Read data set X from the previously selected index. The system 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 system can return the true data of the index from an internal buffer. Being too early with the request command will cause the system 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 simulation start the system will calculate the first PV curve. This takes about 500 ms. But the first data set is already recorded 100 ms after simulation start, so the first 3-4 data set are erroneous. This won't be the case if the recording is started at least 500 ms after the simulation start.

## 5.16.4 Status commands

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
<code>FUNCTION:PHOTOvoltaics:MPP:VOLTage?</code>	Voltage in the MPP, in V. 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 system voltage.
<code>FUNCTION:PHOTOvoltaics:MPP:CURRent?</code>	Current in the MPP, in A. Can be between 0 and rated current.
<code>FUNCTION:PHOTOvoltaics:MPP:POWer?</code>	Power in the MPP, in A. Can be between 0 and rated power.
<code>FUNCTION:PHOTOvoltaics:STATe?</code>	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 and error during PC curve calculation in simulation modes ET or UI
	ERROR DAY = Simulation didn't due to and error during PC curve calculation in simulation modes DAYET or DAYUI
	ERROR ALARM = Simulation has stopped due to a system alarm
	ERROR INTERPOLATION = Simulation not started due to a wrong dwell time value in day trend data index 1

Command	Description
<code>FUNCTION:PHOTOvoltaics:DAY:NUMBER?</code>	Number of accepted day trend indexes. When transferring day trend data to the system, this counter counts up every time an index is successfully transferred and accepted. Can be used to verify written data.
<code>FUNCTION:PHOTOvoltaics:OCVoltage?</code>	Open circuit voltage of the simulated solar panel, calculated with a formula according to the standard. The value is affected by the selected simulation mode, the standard panel parameters (see below) and the calculation factors (also see below).
<code>FUNCTION:PHOTOvoltaics:SCCurrent?</code>	Short-circuit current of the simulated solar panel, calculated with a formula according to the standard. The value is affected by the selected simulation mode, the standard panel parameters (see below) and the calculation factors (also see below).

### 5.16.5 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. When trying to write values it's important what simulation mode (ET, UI, DAYET, DAYUI) and what input mode (MPP, ULIK) is currently set. The matrix below indicates what command 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 modes:				
	MPP	ULIK	MAN	CSI	THIN
<code>FUNCTION:PHOTOvoltaics:FACTOR:FFU &lt;Nrf&gt;</code> <code>FUNCTION:PHOTOvoltaics:FACTOR:FFU?</code> Fill factor for voltage ( $FF_U$ ). Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: $>0-1$	Y	Y	Y	—	—
<code>FUNCTION:PHOTOvoltaics:FACTOR:FFI &lt;Nrf&gt;</code> <code>FUNCTION:PHOTOvoltaics:FACTOR:FFI?</code> Fill factor for current ( $FF_I$ ). Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: $>0-1$	Y	Y	Y	—	—
<code>FUNCTION:PHOTOvoltaics:FACTOR:ALPHA &lt;Nrf&gt;</code> <code>FUNCTION:PHOTOvoltaics:FACTOR:ALPHA?</code> Temperature coefficient $\alpha$ (in $1/^\circ\text{C}$ ) for the short-circuit current. Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: $>0-1$	Y	Y	Y	—	—
<code>FUNCTION:PHOTOvoltaics:FACTOR:BETA &lt;Nrf&gt;</code> <code>FUNCTION:PHOTOvoltaics:FACTOR:BETA?</code> Temperature coefficient $\beta$ (in $1/^\circ\text{C}$ ) for the open circuit voltage. Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: $-1...<0$	Y	Y	Y	—	—
<code>FUNCTION:PHOTOvoltaics:FACTOR:CU &lt;Nrf&gt;</code> <code>FUNCTION:PHOTOvoltaics:FACTOR:CU?</code> Scaling factor $C_U$ for the open circuit voltage. Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: $>0-1$	Y	Y	Y	—	—

Command	Writable in modes:				
	MPP	ULIK	MAN	CSI	THIN
<b>FUNCTION:PHOTOvoltaics:FACTOR:CR &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:FACTOR:CR?</b> Scaling factor $C_R$ in $m^2/W$ or the open circuit voltage. Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: >0-1	Y	Y	Y	—	—
<b>FUNCTION:PHOTOvoltaics:FACTOR:CG &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:FACTOR:CG?</b> Scaling factor $C_G$ in $W/m^2$ for the open circuit voltage. Only writable in technology selection MAN. Has impact on the PV curve calculation with formula according to standard. Range: >0-1	Y	Y	Y	—	—
<b>FUNCTION:PHOTOvoltaics:STANDARD:OCVOLTage &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:STANDARD:OCVOLTage?</b> $U_{oc}$ (open circuit voltage) of the simulated solar panel in V. Range: 0 - rated voltage	—	Y	Y	Y	Y
<b>FUNCTION:PHOTOvoltaics:STANDARD:SCCURRENT &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:STANDARD:SCCURRENT?</b> $I_{sc}$ (short-circuit current) of the simulated solar panel in A. Range: 0 - rated current	—	Y	Y	Y	Y
<b>FUNCTION:PHOTOvoltaics:STANDARD:MPP:VOLTage &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:STANDARD:MPP:VOLTage?</b> Voltage in the MPP of the simulated solar panel, in V. Range: 0 - rated voltage	Y	—	Y	Y	Y
<b>FUNCTION:PHOTOvoltaics:STANDARD:MPP:CURRENT &lt;Nrf&gt;</b> <b>FUNCTION:PHOTOvoltaics:STANDARD:MPP:CURRENT?</b> Current in the MPP of the simulated solar panel, in A. Range: 0 - rated current	Y	—	Y	Y	Y

## 5.16.6 Control commands

These commands are to control the PV simulation, usually after the successful configuration. In some modes, one or two parameters are adjustable while the simulation is running. Any change of parameter requires to calculate the PV curve again, overwriting the former curve. Depending on what point on the curve the simulation currently is, the point will shift after a certain calculation and response time.

Command	Description
<b>FUNCTION:PHOTOvoltaics:STATE {RUN STOP}</b> <b>FUNCTION:PHOTOvoltaics:STATE?</b>	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
<b>FUNCTION:PHOTOvoltaics:TEMPERature &lt;NR1&gt;</b> <b>FUNCTION:PHOTOvoltaics:TEMPERature?</b>	Only available in mode ET: Solar module temperature in °C. Range: -40...+80
<b>FUNCTION:PHOTOvoltaics:IRRAditation &lt;NR1&gt;</b> <b>FUNCTION:PHOTOvoltaics:IRRAditation?</b>	Only available in mode ET: Irradiation in $W/m^2$ . Range: 0-1500

## 5.16.7 Error situations

An error situation occurs 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 5.16.4) can help in both cases to identify the cause of the error.

The following generally applies:

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

## 5.17 Alarm management commands

In remote control operation it's important to manage alarms correctly. This can be done the same way as in manual control. When using the SCPI command language, system alarms are indicated through a status register which can be polled. Polling for errors should be performance after making changes to settings. Furthermore, most alarms have to be acknowledged.

### 5.17.1 Reading system alarms

Reading system alarms should happen in certain intervals, by querying the Questionable status register by either the subregister CONDITION or EVENT. The commands STAT:QUES:COND?, STAT:QUES?, or STAT:QUES:EVENT? return a value that represents certain bits (see "5.5 Status registers" on page 42), indicating various conditions. When a bit is set, it means a certain alarm is present. Refer to the system's operating manual for details about system alarms.

### 5.17.2 Acknowledging system alarms

In order to make the user take notice of system alarms, they have to be acknowledged after they occurred and vanished again. This will delete those alarms from the status register and should be only be done after they have been recorded. To delete/acknowledge an alarm, the command SYST:ERR? resp. SYST:ERR:ALL? is used, which also serves to query other errors.

In case one or multiple alarms are still present, they won't be cleared from the register.

There is one exception in handling, the OT (overtemperature) error. This doesn't require extra acknowledgement and thus won't be indicated anymore in CONDITION once it's gone.

### 5.17.3 Alarm counters

These counters count alarm occurrences since the last time the system was powered. They can be read by command anytime, are not stored when the system is switched off and are purged by reading.

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)

### 5.17.4 Example

You are running the system in remote control and poll the alarm status with STAT:QUES:COND? at a certain interval, and you always receive value 3072. This is the sum of the bit values of bits 10 (remote) and 11 (output on). It tells you that remote control is active and the DC output is switched on. Then a system alarm occurs caused by the unit overheating. When reading the register the next time, bit 3 should indicate the OT alarm for you to take notice. Additionally, the DC output might be indicated as switched off. Thus the returned value could be 1032 or 3080.

---

## 5.18 Example applications

### 5.18.1 Configure and control master-slave with SCPI

Available on 310 and 320 Series.

In a master-slave (MS) setup, usually only the master unit is remotely controlled, while the slaves are usually not connected to the PC, except for when being configured remotely as a slave. It's therefore recommended to configure the MS system on the control panels of the units, and only put the master into remote control via 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 init from the master. The initialization of the MS system is done automatically every time the master is powered, but can be triggered and repeated by command.

Let's assume for an example configuration that we have five power supplies (80 V, 510 A, 15 kW) in parallel. The master has to display itself as an 80V, 2550 A, 75 kW and 1  $\Omega$  unit after successful configuration and initialization. These values are also the temporary new ratings of the master-slave 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 range of 0–110%.

#### ► Part 1a: Configure the master

1. Activate remote control: `SYST:LOCK ON`
2. Activate master-slave mode: `SYST:MS:ENABLE ON`
3. Define the unit as master: `SYST:MS:LINK MASTER`

#### ► Part 1b: Configure the slave(s), if they're connected to the controlling unit (PC, PLC etc.)

4. Activate remote control: `SYST:LOCK ON`
5. Activate master-slave mode: `SYST:MS:ENABLE ON`
6. Define the unit as slave: `SYST:MS:LINK SLAVE`

If there is more than one slave, repeat these steps for each slave with their own addresses.

#### ► Part 2: Initialize the MS system

7. Ensure remote control is activated: `SYST:LOCK ON`
8. Trigger initialization, then wait a few seconds: `SYST:MS:INIT`

#### ► Part 3: Additional optional steps

9. Query the initialization status from the master, in order to analyse it: `SYST:MS:COND?`
10. Query the number of units Initialized for the MS system (should be 5 with this example): `SYST:MS:UNIT?`
11. Query the nominal current of the MS system: `SYST:NOM:CURR?`
12. Query the nominal power of the MS system: `SYST:NOM:POW?`
13. Query the maximum resistance of the MS system: `SYST:NOM:RES:MAX?`
14. Query the minimum resistance of the MS system: `SYST:NOM:RES:MIN?`
15. Configure protection values, for example OCP: `CURR:PROT 400`
16. Configure events, for example,
  - 16.1. set OCD to 2100 A: `SYST:CONF:OCD 2100`
  - 16.2. 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 init, 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
  - 17.3. First, set the current value down to anything lower than the desired limit, like the minimum: **CURR MIN**
  - 17.4. 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 output.

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

The system will remain configured and keep the settings after power-cycling it. The master unit has to initialize the MS and the slaves at least one time after power-up. The status of the first automatic initialization can be read from the master by custom software and depending on the result, the software could trigger further steps like the ones above.

## 5.18.2 Programming examples for PV simulation (DIN EN 50530)

Available on 310 and 320 Series.

Additional information about this extended PV function can be found in the Operating Guide of your system. The user manuals also have information about the connection between simulation mode, input mode, and panel technology.



Important: after the simulation start, the system 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):

Simulation mode \ Options	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

### 5.18.2.1 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:MODE 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 ENABLE	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 values for the standard test condition (STC), 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. This 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/m<sup>2</sup> (E = from German Einstrahlung meaning 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/m <sup>2</sup>	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:DATA?	Read data from data set (index) 1	12024
...	...	Read further n-1 data sets:	...

Nr.	Command	Description	Register
x	FUNC:PHOT:REC:IND n	Select data set n (index n) for reading	12022
y	FUNC:PHOT:REC:DATA?	Read data from data set (index) n	12024

### 5.18.2.2 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:MODE 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 (FF <sub>U</sub> ): 0,8	12034
5	FUNC:PHOT:FACT:FFI 0.78	Fill factor current (FF <sub>I</sub> ): 0,78	12036
6	FUNC:PHOT:FACT:ALPH 0.0003	Temperature coefficient $\alpha$ for I <sub>sc</sub> : 0,0003 / °C	12038
7	FUNC:PHOT:FACT:BETA -0.003	Temperature coefficient $\beta$ for U <sub>oc</sub> : -0,003 / °C	12040
8	FUNC:PHOT:FACT:CU 0.0725	Scaling factor C <sub>U</sub> for U <sub>oc</sub> : 0,0725	12042
9	FUNC:PHOT:FACT:CR 0.00022	Scaling factor C <sub>R</sub> for U <sub>oc</sub> : 0,00022 m <sup>2</sup> /W	12044
10	FUNC:PHOT:FACT:CG 0.00315	Scaling factor C <sub>G</sub> for U <sub>oc</sub> : 0,00315 W/m <sup>2</sup>	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 ENABLE	Activate data recording	12018
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 $\geq U_{oc}$ )	500



- The values for the standard test condition (STC), 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 system to calculate the PV anew.
- Voltage and current in the MPP are connected via factors FFI and FFU to the open circuit voltage (U<sub>oc</sub>) and the short-circuit current (I<sub>sc</sub>), 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:MODE WRITE	Select access mode: write	12006
19	FUNC:PHOT:DAY CLEAR	Delete former data (should be executed every time before loading new data)	12007
20	FUNC:PHOT:DAY:DATA 1, 500, 20, 1500	Write 1 <sup>st</sup> day trend data set: Irradiation: 500 W/m <sup>2</sup> 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:DATA 2, 800, 28, 1500	Write 2 <sup>nd</sup> day trend data set: Irradiation: 800 W/m <sup>2</sup> Temperature: 28 °C Dwell time: 1500 ms	12010
...	...	Write further data sets, a total of 500	...
519	FUNC:PHOT:DAY:DATA 500, 1200, 35, 20000	Write 500. day trend data set: Irradiation: 1200 W/m <sup>2</sup> 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 is not 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:DATA?	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:DATA?	Read data from data set (index) n	12024

### 5.18.2.3 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:MODE 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 DISABLE	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 $\geq U_{oc}$ )	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

### 5.18.2.4 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:MODE 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

Nr.	Command	Description	Register
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 DISABLE	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 $\geq U_{oc}$ )	500

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

Nr.	Command	Description	Register
11	FUNC:PHOT:DAY:MODE WRITE	Select access mode: write	12006
12	FUNC:PHOT:DAY CLEAR	Delete former data (should be executed every time before loading new data)	12007
13	FUNC:PHOT:DAY:DATA 1, 1, 1, 300000	Write 1 <sup>st</sup> day trend data set: MPP voltage: 1 V MPP current: 1 A Dwell time: 300 seconds -> 5 minutes	12010
14	FUNC:PHOT:DAY:DATA 2, 2, 2, 500	Write 2 <sup>nd</sup> day trend data set: MPP voltage: 2 V MPP current: 2 A	12010
...	...	Write further data set, a total of 1000	...
1012	FUNC:PHOT:DAY:DATA 1000, 30, 9, 500	Write 1000th day trend data set: 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

# 6 Profibus & Profinet

## 6.1 General

Connection to Profibus and Profinet are possible only through the Anybus modules IF-AB-PBUS (Profibus) or IF-AB-PNET (Profinet, 1 or 2 ports), and therefore limited to the 310 series of systems.

The Profinet/IO module (1 or 2 ports) can be used to control and monitor a system using a network system, usually combined with an integrated PLC and proper software. For Profinet, the software selects the necessary Ethernet port, because this port cannot be adjusted on the system. The standard Profinet communication is handled by the field bus protocol via special software.

On the system side, the interface module simplifies the necessary configuration. When using Profibus, the user only has to set a slave address (0–125), while Profinet’s network settings are usually configured through remote control, often using the Siemens Primary Setup Tool (PST) or something similar. Optional parameters like tags can be defined in the system’s setup menu or via command.

This part of the document covers how to use the ModBus register lists (PDF) with your system in order to access indexes and slots via acyclic communication. The interface modules present the system as a DP-V1 slave to the network, capable of cyclic and acyclic data transmission.

## 6.2 Preparation

For the implementation of a system into 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 system description file called GSD (Generic Station System) for Profibus, or a GSDML for Profinet, which is available as a download from our website.

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

## 6.3 Slot configuration for Profibus

The slot configuration for users of the interface module IF-AB-PBUS is done by loading the GSD/GSE file in the configuration dialogue (with Siemens STEP7: HWCONFIG) and by arranging the slots in a specific order:

Slot	Slot name	Description
1	System status & acyclic slot 1	Cyclic: System status (see register list) Acyclic: all registers (indexes) assigned to slot 1
2	Act. voltage & acyclic slot 2	Cyclic: Actual voltage of DC input/output Acyclic: all registers (indexes) assigned to slot 2
3	Act. current & acyclic slot 3	Cyclic: Actual current of DC input/output Acyclic: all registers (indexes) assigned to slot 3
4	Act. power & acyclic slot 4	Cyclic: Actual power of DC input/output Acyclic: all registers (indexes) assigned to slot 4
5	Acyclic slot 5	Acyclic: all registers (indexes) assigned to slot 5
...	...	...
12	Acyclic slot 12	Acyclic: all registers (indexes) assigned to slot 12

Example view of the HW CONFIG in Siemens Simatic, with 8 slots:

Steckplatz	DP-Kennung	Bestellnummer / Bezeichnung	E-Adresse	A-Adresse
1	209	Status and acyclic slot 1	256...259	
2	208	Act. voltage & acyclic slot 2	260...261	
3	208	Act. current & acyclic slot 3	262...263	
4	208	Act. power & acyclic slot 4	264...265	
5	208	Acyclic slot 5	266...267	
6	208	Acyclic slot 6	268...269	
7	208	Acyclic slot 7	270...271	
8	208	Acyclic slot 8	272...273	

The address ranges can of course be re-arranged as required. The slots for acyclic access don't need an output address range, because the input address range already reserves memory space for both directions.

## 6.4 Slot configuration for Profinet

The GSDML file does not 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:

Slot	Slot name	Description
1	Input 2 words	Cyclic: System status (register 505, see register list) Acyclic: all registers (indexes) assigned to slot 1
2	Input 1 word	Cyclic: Actual voltage of DC input/output (register 507, see register list) Acyclic: all registers (indexes) assigned to slot 2
3	Input 1 word	Cyclic: Actual current of DC input/output (register 508, see register list) Acyclic: all registers (indexes) assigned to slot 3
4	Input 1 word	Cyclic: Actual power of DC input/output (register 509, see register list) Acyclic: all registers (indexes) assigned to slot 4
5	Input 1 word	Acyclic: all registers (indexes) assigned to slot 5
...	...	...
12	Input 1 word	Acyclic: all registers (indexes) assigned to slot 12

Example view of the HW CONFIG in Siemens Simatic, with 8 slots:

Steckplatz	Baugruppe	Bestellnummer	E-Adresse	A-Adresse	Diagnoseadresse
0	Default	ABCC-FPT (2-Port)			2043
1	Input 2 word		256...259		
2	Input 1 word		260...261		
3	Input 1 word		262...263		
4	Input 1 word		268...269		
5	Input 1 word		270...271		
6	Input 1 word		272...273		
7	Input 1 word		274...275		
8	Input 1 word		264...265		
9					

The address ranges can of course be re-arranged as required. The slots for acyclic access don't need an output address range, because the input address range already reserves memory space for both directions.

---

## 6.5 Cyclic communication via Profibus/Profinet

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” resp. “6.4. Slot configuration for Profinet”.

Actual values coming from the system have to be translated to real values according to the formula described in section “4.2.1. Hex Percent and Decimal Value Conversion”, while any other data are referenced in those 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”, a name which can be found in the register list at position 508. This is also where the register is enabled for use with Profibus/Profinet use by having a slot/index number assigned.

According to sections 6.3 and 6.4 there are up to 12 slots for acyclic system access, which carry a varying number of indexes (see register lists). By using appropriate blocks/functions (SFB52, SFB53), the user can acyclically access the IDs (slot addresses) and indexes by write and read. The slots for acyclic transfer are only defined to reserve slot addresses and memory space, so it doesn't matter that they are only inputs.



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

---

## 6.6 Acyclic communication via Profibus/Profinet

Acyclic communication with the target system is done by using slots 1-12, precisely their resulting ID, and indexes, which are accessed by system function blocks for read or write. The SFBs to use here are usually SFB52 and SFB53 when using Siemens software. Other PLC control softwares offer similar options.

The SFBs 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 “4.2.1. Hex Percent and Decimal Value Conversion”.

For starters there are example projects (on the included USB stick or as download), one each for Profibus and Profinet, which can be opened with Siemens STEP7 and which shall demonstrate the access to the system with preconfigured data blocks.

The register list for your system 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 respectively, also in “4.2.1. Hex Percent and Decimal Value Conversion”. Rule of thumb:

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

The general procedure to control a system remotely is like this:

1. Activate remote control with the appropriate command (may be denied by the system, see “3.4. Control Location”)
2. Control and monitor your system remotely, via cyclic (DP-V0) and/or acyclic (DP-V1) access.
3. Deactivate, i.e. leave remote control

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

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

output can't be determined definitely.

In order to send a command from with a typical Profibus/Profinet software, following applies in general:

1. Select the desired register from the register list and read its assigned slot/index values.
2. Determine the I/Q address which is assigned to the particular slot in HWCONFIG. The use of ID, index, slot and subslot are not just different between Profibus and Profinet, but also between the different PLC systems and softwares. The examples below demonstrate the use with Siemens software.
3. Set ID, index and parameter (set value, status or something else) in decimal or hexadecimal form in the SFB and execute.
4. Process the data returned from the system, if the last message was a query.

---

## 6.7 Examples for acyclic access

### 6.7.1 Activate/deactivate remote control

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

#### ► Activating or deactivating remote control of your system via Profibus

1. Use the register list and find the proper command, here: Register 402 - Remote mode.
2. Find the slot and index values for this command in the dedicated columns, here slot 2 and index 1.
3. From the slot configuration read the I/Q address for slot 2 to have the value for parameter "ID", for example 260 (like in the example configs in 6.3 and 6.4) or DW#16#104
4. The value "Index" from the register list is submitted to the parameter INDEX like this:  
Profibus: INDEX = Index = 1  
Profinet: INDEX = Slot number \* 255 + 1 + Index = 510 + 1 + 1 = 512
5. Use a suitable function block in your automation software, for example SFB53.
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. Configure the function block with ID, INDEX and control value and execute the block. If not somehow inhibited by the system, it should either switch to remote control or back to manual control.

### 6.7.2 Send a set value

Any command that sets something in the system, no matter if value or status, requires activated remote control status. Also see "6.7.1. Activate/deactivate remote control" and "3.4. Control Location".

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 percent of the nominal values. Read sections "4.2. Set Value Resolution" and "4.2.1. Hex Percent and Decimal Value Conversion" for more information.

#### ► Setting the DC 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 command in the dedicated columns, here slot 2 and index 24.
3. From the slot configuration read the I/Q address for slot 2 to have the value for parameter "ID", for example 260 (like in the example configs in 6.3 and 6.4) or DW#16#104

4. The value "Index" from the register list is submitted to the parameter INDEX like this:  
 Profibus: INDEX = Index = 25  
 Profinet: INDEX = Slot number \* 255 + 1 + Index = 510 + 1 + 24 = 535
5. Use a suitable function block in your automation software, for example SFB53.
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 the function block and execute the block. The system should instantly set 10 A as current limit. This can be verified in the display of the system where it shows the set value of current.

### 6.7.3 Read something

Reading something from the system 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 acyclic transfer.

#### ► Reading the actuals 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 of voltage is the lower number, thus it will be: Register 507 - Actual voltage
2. Find the slot and index values for this command in the dedicated columns, here slot 2 and index 28
3. From the slot configuration read the I/Q address for slot 2 to have the value for parameter "ID", for example 260 (like in the example configs in 6.3 and 6.4) or DW#16#104.
4. The value "Index" from the register list is submitted to the parameter INDEX like this:  
 Profibus: INDEX = Index = 28  
 Profinet: INDEX = Slot number \* 255 + 1 + Index = 510 + 1 + 28 = 539
5. 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.
6. Use a suitable function block in your automation software, for example SFB52.
7. Configure the function block with ID, INDEX and data length (4 bytes or 2 words of 16 bit, depending in the way the software defines the input).
8. 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 percent value (for translation see "4.2.1. Hex Percent and Decimal Value Conversion"). 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.

## 6.8 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 "4.2.1. Hex Percent and Decimal Value Conversion".

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: System status (connected to register 505). The value 0x000004C0 says that bits 6, 7 and 10 are set. It means, the system 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  $250 \text{ V} * 0x263A / 52428 = 46.7 \text{ V}$ .

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

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

# 7 CANopen

The available communication objects (ADIs) in an Electronic Data Sheet file (EDS/XDD) are available from our website. This EDS can be integrated in CANopen related software. The EDS indexes are not separately explained, because their definition and use is identical to the ModBus register list files (see “4.5. Reading 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 users are not confronted with checksums and function codes as with ModBus.



The CANopen module IF-AB-CANO does not feature an internal termination resistor. Thus the bus termination has to be applied by the user according to the CAN bus requirements.

## 7.1 Preparation

For the communication with the system 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 activated always if the system is at the end of the bus, like when directly connecting the PC to a single unit.
2. EDS/XDD (included with the system on USB stick).
3. CANopen software for the PC (not included, 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).

## 7.2 User objects (indexes)

The message format used via CANopen communication is related to ModBus. A specific index is connected to a specific ModBus register. The CANopen standard defines that user objects are enumerated from index 0x2001. With ModBus, the registers are counted from 0. This means, that index 0x2001 corresponds to register 0, and index 21F5 corresponds to register 0x01F4 (decimal 500) etc.

The EDS/XDD contains fewer indexes than mPower supports for ModBus registers, but the available indexes still cover most functions needed for remote control of an mPower system. Users can edit the EDS/XDD anytime, and add indexes.

Along with this document, there are ModBus register lists. These can also be used to CANopen, as they also define data type and value range of the indexes. Examples in other sections of this documents can be applied to CANopen as well.

### 7.2.1 Translation ADI > Register

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

Index number - Offset = register address → 0x207A - 0x2001 = 0x79 (hex) = 121 (dec). According to the register list for an mPower 310 Series, this represents the nominal system voltage as a FLOAT value. Since CANopen does not support the data type FLOAT, the EDS uses REAL32 here. The user just has to translate the 32 bit value according to IEEE 754 specification.

---

## 7.3 Specific examples

### 7.3.1.1 Switching to remote control

As described in “4.6.9.5. Switch between remote and manual control”, it’s required to switch the system to remote control before you can control it. In order to do this, you first need to find the proper command in the register list relative to the dedicated index in the EDS. In this case, it’s register 402 which is 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.

### 7.3.1.2 Setting a set value

After remote control has been accepted by the system, you are allowed to send set values. Those values usually represent a percent value. From the definition in the register list, 100% of a value translated to the hexadecimal value 0xCCCC and 0% to 0x0000. There are 52429 possible values between 0% and 100%. It has to be pointed out here, that this is not the true resolution values like voltage or current actually achieve at the DC output. The effective resolution of output/input values is 26214 steps. More details are available in “4.6.9.1. Writing a set value”.

---

## 7.4 CANopen to ModBus differences

### 7.4.1 When using the arbitrary generator

Due to CANopen only being able to transport a maximum of 4 effective user data bytes per message, the 8 values of data defining a sequence point of the arbitrary generator cannot be transferred at once. Instead, they are sent in 8 separate messages. The system checks every single value for plausibility upon reception, but once all sequence points are set without any error, an additional submit command is needed (index 235F). This will transfer all sequence point data and load the function into the function generator, and enable start/stop action. Without sending the submit command, the function generator would either run with all data being zero or using old data.

The steps to perform, as described in section 4.9.9.1, are the same for CANopen, except for the additional step 3.1:

Step 1:

Select, whether to apply the function to the voltage U (index 2354) or the current I (index 2355). Before making this selection, the system cannot accept sequence point data, because the data is run through a plausibility check against the system’s nominal values.

Step 2:

Define start sequence point (index 235C), end sequence point (index 235D) and number of cycles of that sequence point block to repeat (index 861).

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, undocumented for ModBus, because not required there)

Step 4:

Set global voltage limit (index 21F5), if the function is applied to the current. Otherwise, set global current limit (index 21F6), if the function is applied to voltage. Set global power limit (index 21F7) for both modes.

Step 5:

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

Step 6:

When finished, leave the function generator by deselecting your former selection of either U (index 2354) or I (index 2355) again by writing 0x0000.

---

## 7.5 Error codes

Following error codes, as part of the CANopen standard, are supported by the CANopen interface module:

Code	Description
0x06020000	Object does not exist in the object dictionary (ModBus register list)
0x06040043	Command not supported
0x06099911	Sub-Index does not 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 system state
0x05040005	Out of memory
0x08000000	General error

This section is specific to the communication with a system via the CAN Anybus module IF-AB-CAN—and therefore is specific to the 310 Series systems. Configuration of the interface itself is done on the control panel (HMI) of the system.

---

## 8.1 Preparation

Requirements for communication with the system via CAN module IF-AB-CAN.

1. A suitable CAN cable. It's not required to have one with integrated bus termination switch and resistor, because the interface module has an electronically switched resistor for bus termination. In case the cable also has one, it's important to take care to activate only one of both, otherwise there can be bus errors.
2. When using Vector™ or similar software which can make use of database files (DBC), a dedicated DBC for the particular system model is needed. If not available, it can be created by the user, for example by modifying a similar one.
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 register list for the series.

---

## 8.2 Introduction

The data format is derived from ModBus RTU. In relation to a database file (DBC) a mux value (Vector terminology) represents a specific ModBus register or object/command. Objects in the database are selected by the muxer, and when programming the CAN message buffer directly (CAPL), the first two bytes of data in a CAN message define the register (object, command) to access. The selection between writing and reading objects is done by the CAN ID.

Each system will be assigned at least three CAN IDs, which are defined via the Base ID in the system's CAN settings. The Base ID is used write to objects (message type: Send\_Object), while querying objects (message type: Query\_Object) is done with Base ID+1 and responses (message type: Read\_Object) coming from the system use Base ID+2. Responses from the system are expected after a query, but can also be received unexpectedly in case of communication or access error. When adjusting the Base ID of a system, the other 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 systems at once by one command when using the same broadcast ID on these system. This ID is for write access (Send\_Object) only. Queries to multiple systems at once with one message are not possible.

Apart from the Base ID and Broadcast ID for acyclic access, some series support further adjustable IDs for cyclic status data which is sent permanently by the system once activated and after the CAN connection has been established. Refer to the system manual for setup, particularly the section for the communication settings, and also further below.

## 8.3 Message formats



Below explanations are, besides the selection of IDs to switch between write and read actions, also related to the ModBus functions, as listed in the register lists in columns 2-6.

### 8.3.1 Normal sending (writing)

Writing to the system always used the base ID or the broadcast ID. It requires defining the first register/object to write to in the CAN data, as well as the number of registers to write and a specific number of parameter bytes which can represent different data types.

Access: Base ID, broadcast ID

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

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

Access: Base ID, broadcast ID

Connected ModBus function: Write Multiple Registers (WMR)

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 the register number from the register list, i. e. start register, even for WMR.

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: is used to distinguish single messages from split messages and to detect the correct sequence of data. For example, a string like the user text can be up to 40 characters long 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 CAN bus it's not guaranteed that messages are received in the same order they were sent.

Data bytes: the number of bytes in this type of message is always 4, no matter if all bytes are filled with information from the actual data to transmit or are 0. An example: an user text with a length of 15 characters would require to send at least 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 (15 bytes of string + termination character).

### 8.3.2 Cyclic sending (writing)

Cyclic sending (or writing) is intended for compact and time efficient transmission of often used set values and status in form of a block of data. It uses separate CAN IDs. The user defines the interval of cyclic send via timing in the CAN software, though we recommend to stick to timing recommendations as described in section 3.5.3.



It's strongly recommended to send the block **Control** after the other data blocks, especially when remote control has not yet been activated. This means you should write the set values first, and as the last step write the status to **Base ID Send**. This will submit all data internally and since block **Control** would contain **remote control = on**, the set values in the other blocks would be accepted without error.

In order to use this feature, the user only has to define the separately adjustable **Base ID Cyclic Send** and can then send two different messages with following format:

Access: Base ID Cyclic Send (Control)

Bytes 0-1
Control word

Control word definition:

Bit	Name	Related register	Meaning
0	Remote control	402	Activates remote control of the system with 1 or deactivates it with 0
1	Input/Output	405	Switches the DC output of the system on with 1 or off with 0
2	UIP / UIR	409	Activates resistance control mode (UIR, where featured) with 1, while with 0 mode UIP will be active
3	(not applicable)	422	—
4	Alarm	411	The value 1 acknowledges all currently acknowledgeable alarms



This control word requires special attention, as the 5 bits can trigger several actions at once which don't have a certain priority of processing. This means that if you would try to activate remote control together with switching on the DC output (bits 0 and 1 both TRUE), you may receive a settings conflict error, because the system would possibly process bit 1 before bit 0.

Access: Base ID Cyclic Send + 1 (Set values 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

### 8.3.3 Querying

Querying an object is the first part of a read action. It's always done via Base ID + 1. The system should then respond via Base ID + 2 (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 system will respond with the correct length of data, but in different. See below at [8.3.4](#).

Access: Base ID + 1

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

Bytes 0+1
Start reg.
0-65534

### 8.3.4 Normal reading

Data coming from the system can be a single message (expected data or error) or can be split messages forming a response. The information is either in a buffer or, when using Vector software, automatically sorted into signals. The data of split messages has to be combined again according to the marker. Even the Vector database cannot do this automatically. But there are only a few objects like the user text which require this treatment and these are usually not accessed very often.

Access: Base ID + 2

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

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

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

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

Response as error message:

Bytes 0+1	Byte 2
65535	Error code

The error codes used here are the same as with ModBus. See "[4.6.8. Communication errors](#)".

## 8.3.5 Cyclic reading

The cyclic read feature is an extended function where the system can automatically send specific objects to specific IDs and in a specific interval. Cyclic read messages differ from those of normal read actions.

In order to activate and use cyclic read, the user has to

1. Set the extra **Base ID Cyclic Read** on the system (HMI, CAN settings).
2. Define which of the 5 available objects for cyclic read are going to be used and activate them by setting the interval time to a value other than zero.
3. Process the received data separately and differently, because the data format is different here (see below).

The interval times for the cyclic objects can be set separately and arbitrarily. In case they match or overlap, the system will send the corresponding messages subsequently and as fast 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 read items are active, because of too much traffic.

Once cyclic read is activated by setting the interval time of at least one item and as soon as a CAN connection is established, the system will start to automatically and permanently send messages to the defined IDs. The cyclic read feature can be turned off or on anytime using the CAN settings on the control panel or the corresponding commands sent as acyclic CAN message.

There are up to 6 CAN IDs reserved for cyclic read. Starting at the adjustable **Base ID Cyclic Read** (see control panel of the system) the data in the messages is defined as follows:

Access: Base ID Cyclic Read (Status)

Bytes 0-3
System status (32 Bit)

Bit layout of the system status value:

Bit	Name	Meaning
31	Remote control	1 = on
30	Input / output	1 = on (req., register 405)
29	Volt. reg. speed	1 = fast (register 422)
28	Operation mode	0 = UIR, 1 = UIP
27	Alarms	1 = at least 1 alarm active
26	Alarm MSS	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

Bit	Name	Meaning
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 = 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	Reg. mode	register 505, bits 10-9
1		
0	PSB mode	0 = source, 1 = sink

Access: Base ID Cyclic Read + 1 ("Actual values")

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

Access: Base ID Cyclic Read + 2 ("Set values 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

Access: 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

Access: 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

Access: Base ID Cyclic Read + 5 ("Set values [EL]") (PSB 9000 and PSB 10000 series only)

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)

Access: Base ID Cyclic Read + 6 ("Limits [EL]") (PSB 9000 and PSB 10000 series only)

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 Message examples

### 8.3.6.1 Switching to remote control

As described in "4.6.9.5. Switch between remote and manual control", it's required to switch the system 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 resp. the dedicated index in the EDS. 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 system has been set to Base ID 0x20, the data to be sent according to 8.3.1 would be:

0x01	0x92	0x01	0xFF	0x00
Register / object	Nr. of regs	Bit (coil) for TRUE		

The system should switch to remote control immediately after reception, if not inhibited somehow. The status of remote control can be read from the display or by reading another object.

### 8.3.6.2 Write and read back a set value

After remote control has been accepted by the system, you are allowed to send set values. Those values usually represent a percent value. From the definition in the register list, the hexadecimal value 0xCCCC translates to 100% and 0x0000 to 0%. It means, there are 52428 possible values between 0% and 100%. It has to be pointed out here, that this is not the resolution a system value like voltage or current can have at the DC input/output. The effective resolution of output/input values is 26214 steps. An example for set value translation is in “4.6.9.1. Writing a set value”.

Power supply model PSI 9080-170 3U has a nominal current of 170 A. If you wanted to set it to 35 A, the set value according to the formula in 4.2.1 calculates as:  $35 \text{ A} * 52428 / 170 \text{ A} = 10794 = 0x2A2A$ . The current is set with register 501. Assuming the system would have been set to Base ID 0x88 the data to be sent to this ID, according to 8.3.1, would be:

0x01	0xF5	0x01	0x2A	0x2A
Register / object		Nr. of regs	Bit (coil) for TRUE	

Soon after the system 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. With the same base ID, the query message would be

0x01	0xF5
Register / object	

and would have to be sent to the query ID of the system, here 0x89. Shortly after this, the system should respond the requested value on the read ID 0x8A:

0x01	0xF5	0x2A	0x2A
Register / object		Set value current	

In case the values has not been accepted when sending it, for example because the adjustment limit for current (I-max) has been set to 30 A, the system may have responded with an error message (see 8.3.4) instead of the expected one:

0xFF	0xFF	0x03
Error	Error	Code

The ModBus error code 0x3 indicates wrong data. In this case, the set value was too high.

# 9 EtherCAT

## 9.1 Preamble

This section is specific to the communication with a system via the Anybus module IF-AB-ECT—and therefore is specific to the 310 Series systems.

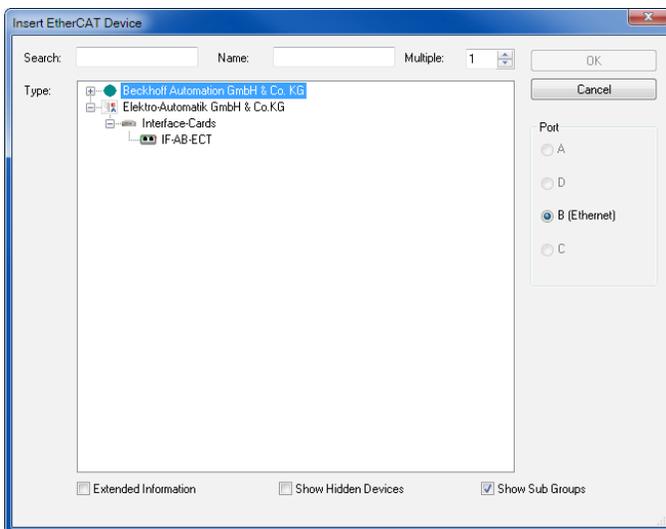
The EtherCAT data communication is based on CANopen. All documentation for EtherCAT and CANopen is provided by the Beckhoff company and the CAN in Automation (CiA) organization. In the documentation below, certain software examples refer to Beckhoff's TwinCAT.

## 9.2 Integrating your system in TwinCAT

To use EtherCAT, you will need an ESI file (an EtherCAT system description in XML format) from our website. The file is to be put into a dedicated folder in the TwinCAT installation. Default path on Windows system:

```
c:\TwinCAT\\Config\Io\EtherCAT\
```

After installing that file and restarting the TwinCAT IDE, mPower EtherCAT slaves can be integrated into the setup with the Insert EtherCAT System dialog and by selecting the system name IF-AB-ECT.



## 9.3 Data objects

The systems internally use ModBus protocol and for CANopen over Ethernet communication. This is why the reference for all cyclic data (PDOs) and acyclic data (SDOs) is the ModBus register lists (“4.5. Reading Register Lists”). The acyclic objects are downloaded from the system when accessing an online EtherCAT slave in tab "CoE" in TwinCAT. Offline objects in form of an EDS file are not available.

Together with the PDOs defined in the ESI file the complete list of indexes then becomes accessible and enable the user to completely control the system.

There is a connection between the CoE indexes and the ModBus register numbers in the lists. You can translate both back and forth.

## Translating ModBus register to CANopen index

ModBus register number in decimal + 8193 => convert to hexadecimal = index

Example: you want to set the system into remote control mode and want to find the corresponding CoE index. In the register list you have register number 402 for this task. The calculation is:  $402 + 8193 = 8595$  => converted to hexadecimal is 0x2193, and thus the index is 2193.

## Translating CANopen index to ModBus register

CANopen index in hexadecimal - 0x2001 => convert to decimal = register

Example: you need know the meaning of the bits in the “Status” PDO. Find the corresponding CoE index in the index list. In this case, it's 21FA. The calculation is:  $0x21FA - 0x2001 = 0x1F9$  => converted to decimal is 505. In the register list you will find register number 505 and the layout of the 32 bit value.

### 9.3.1 PDO object

The system description file defines for mPower EtherCAT slaves the same set of object in the one PDO:

Name	EtherCAT data type	Length in bytes	ModBus register	Short description
Status	UDINT	4	505	System status
Voltage Monitor	UINT	2	507	Actual voltage on DC output (in percent)
Current Monitor	UINT	2	508	Actual current on DC output (in percent)
Voltage select	UINT	2	500	Set value of voltage (in percent)
Current select	UINT	2	501	Set value of current (in percent)
Power select	UINT	2	502	Set value of power (in percent)
Resistance select	UINT	2	503	Set value of resistance (in percent)

### 9.3.2 SDOs

The acyclic data objects for use in the EtherCAT system are defined in your system and can be downloaded from it. It requires the system to be online with the EtherCAT system. There is no separate documentation for the downloadable data objects. Like with CANopen (see “7. CANopen”), the register lists which are part of the programming documentation are the reference for the SDOs to explain data content and function, as well the section of this document dealing with the ModBus protocol and its examples.

### 9.3.3 Use of the data objects

Please refer to “7.2. User objects (indexes)”.

# Appendix A: System Classes

## A.1 Class Assignments

For distinction of different system series and especially of variants within one series a system class number is assigned to every system. It can be read from the system (ModBus register 0, or SCPI command SYSTEM:DEVice:CLASS?) and used to distinguish products when scanning a network for mPower units.

Class	Assigned to series
28	mPower 300 Series 2U/3U models
30	mPower 300 Series 1U models
33	mPower 310 Series 2U/3U models
45	mPower 320 Series

# Appendix B: 320 Series Front USB

The main purpose of the front USB port is quick access to the most important DC output related parameters, such as set values and protections. Reading values and status is always possible, while setting them is only possible when the unit is not in control by a master device.

The number of available commands is restricted on the front USB port relative to the rear port, but it supports both ModBus RTU and SCPI protocols.

## B.1 ModBus Commands

The available ModBus RTU commands are identified in a Register List specific to the 320 Series Front USB. This list is separate from the main 310/320 Series Register list document. The file is available from our website.

## B.2 SCPI Commands

In the programming guide there is a section for all SCPI commands, as available with the rear USB port. Here is an overview what commands are available with the front port. Details about the commands can be found in the “Programming SCPI & ModBus” document, also called programming guide.

*IDN?	[SOURce:]VOLTage
*CLS	[SOURce:]VOLTage?
*RST	[SOURce:]VOLTage:LIMit:HIGH?
*ESE	[SOURce:]VOLTage:LIMit:LOW?
*ESE?	[SOURce:]VOLTage:PROTection[:LEVeL]
*ESR	[SOURce:]VOLTage:PROTection[:LEVeL]?
*STB?	MEASure:[SCALar:]CURRent[:DC]?
[SOURce:]CURRent	MEASure:[SCALar:]POWer[:DC]?
[SOURce:]CURRent?	MEASure:[SCALar:]VOLTage[:DC]?
[SOURce:]CURRent:LIMit:HIGH?	OUTPut[:STATe]
[SOURce:]CURRent:LIMit:LOW?	OUTPut[:STATe]?
[SOURce:]CURRent:PROTection[:LEVeL]	STATus:OPERation?
[SOURce:]CURRent:PROTection[:LEVeL]?	STATus:QUEStionable?
[SOURce:]IRRAdiation	SYSTem:ALARm:ACTion:PFAil
[SOURce:]IRRAdiation?	SYSTem:ALARm:ACTion:PFAil?
[SOURce:]POWer	SYSTem:ALARm:COUNt:OCURrent?
[SOURce:]POWer?	SYSTem:ALARm:COUNt:OPOWer?
[SOURce:]POWer:LIMit:HIGH?	SYSTem:ALARm:COUNt:OTEMperature?
[SOURce:]POWer:LIMit:LOW?	SYSTem:ALARm:COUNt:OVOLTage?
[SOURce:]POWer:PROTection[:LEVeL]	SYSTem:ALARm:COUNt:PFAil?
[SOURce:]POWer:PROTection[:LEVeL]?	SYSTem:COMMunicate:TIMEOUT?
[SOURce:]RESistance	SYSTem:CONFig:MODE

[SOURCE:]RESistance?	SYSTEM:CONFIg:MODE?
[SOURCE:]RESistance:LIMit:HIGH?	SYSTEM:CONFIg:OCD
SYSTEM:CONFIg:OCD?	SYSTEM:CONFIg:USER:TEXT?
SYSTEM:CONFIg:OCD:ACTion	SYSTEM:CONFIg:UVD
SYSTEM:CONFIg:OCD:ACTion?	SYSTEM:CONFIg:UVD?
SYSTEM:CONFIg:OPD	SYSTEM:CONFIg:UVD:ACTion
SYSTEM:CONFIg:OPD?	SYSTEM:CONFIg:UVD:ACTion?
SYSTEM:CONFIg:OPD:ACTion	SYSTEM:DEVIce:CLAss?
SYSTEM:CONFIg:OPD:ACTion?	SYSTEM:ERRor:ALL?
SYSTEM:CONFIg:OUTPut:REStore	SYSTEM:ERRor:NEXt?
SYSTEM:CONFIg:OUTPut:REStore?	SYSTEM:ERRor?
SYSTEM:CONFIg:OVD	SYSTEM:LOCK
SYSTEM:CONFIg:OVD?	SYSTEM:LOCK?
SYSTEM:CONFIg:OVD:ACTion	SYSTEM:LOCK:OWNer?
SYSTEM:CONFIg:OVD:ACTion?	SYSTEM:NOMinal:CURRent?
SYSTEM:CONFIg:UCD	SYSTEM:NOMinal:POWer?
SYSTEM:CONFIg:UCD?	SYSTEM:NOMinal:RESistance:MAXimum?
SYSTEM:CONFIg:UCD:ACTion	SYSTEM:NOMinal:RESistance:MINimum?
SYSTEM:CONFIg:UCD:ACTion?	SYSTEM:NOMinal:VOLTage?

mPower™ DC 3xx Series  
Programmable Power Supplies  
Programming Guide P/N: 501048-DC3PS-A

© 2020, Marway Power Systems, Inc. All rights reserved.



Marway Power Solutions  
1721 S. Grand Ave., Santa Ana, CA 92705  
800-462-7929 • [marway@marway.com](mailto:marway@marway.com)