Command Reference Manual

BatReg2

CAEN ELS s.r.l.

v1.0.4 - February, 2025

TABLE OF CONTENTS

CHAPTER

ONE

WELCOME TO THE COMMAND REFERENCE MANUAL

This manual covers the following power supplies families:

- BatReg2 series

CHAPTER

TWO

COMMUNICATION INTERFACE

In this chapter the basic characteristics of the remote communication interface are listed.

2.1 Ethernet Interface

The control interface is based on ASCII commands over the *TCP* or *UDP* protocol on the **port 10001**.

The device default Ethernet parameters are the following:

Parameter	Factory value
IP	192.168.0.10
Subnet mask	255.255.255.0
Gateway	192.168.0.1
DHCP	Disabled

The Ethernet configuration can be modified using the local interface menu.

Tip: The Ethernet Configuration can be changed, but it is not possible to modify the communication protocol port, which remains **10001**.

2.2 Command Syntax

Commands are in *ASCII format* and are **NOT CASE SENSITIVE** and therefore the command string can be sent either using uppercase or lowercase characters.

- Each command consists of one or more fields, the fields are separated by colons `:`
- Each command has to be terminated with the **CRLF** ("Carriage return, line feed") termination sequence `\r\n`

Note:

- In this documentation the commands are listed in **uppercase**.
- To facilitate the reading, **the termination sequence is not included in the code examples**, but it is necessary to include it at the end of every command.

2.2.1 Read Commands

The read commands have the following format:

- *command name*, followed by one or more optional sub-commands separated by colons `:` ;
- colon char `:` ;
- question mark `?` ;
- termination sequence CRLF `\r\n` .

The reply to a read command has the following format:

- hashtag `#` ;
- *command name echo* is the echo of the command that was sent, without the question mark ;
- colon char `:` ;
- *read value* or *read values* separated by colon `:` ;
- termination sequence *CRLF* `\r\n` .

Example(s):

- An example of a *single level* read command is:

```
OUT:?\r\n
#OUT:ON\r\n
```

- An example of a *multi-level* read command is:

```
GET:I:?\r\n
#GET:I:1.0658\r\n
```

2.2.2 Write Commands

The write commands have the following format:

- *command name*, followed by one or more optional sub-commands separated by colons `:` ;
- colon char `:` ;
- *write value* or *write values* separated by colon `:` ;
- termination sequence `\r\n` .

The reply of the power supply to a write command can be:

- `#AK` (*Acknowledged*): when the command has been **accepted**;
- `#NAK` (*Not Acknowledged*): when the command is **NOT accepted**, followed by the corresponding *error code* and the *error description* separated by a space character.

Example(s):

- An example of a write command when it is **accepted**:

```
LOOP:CV\r\n
#AK\r\n
```

- An example of a write command when it is **NOT accepted**:

```
SET:I:2\r\n
#NAK:16 Module is not in ON\r\n
```

Tip: By default the **NAK** (*Not Acknowledged*) reply returns also the error description. The error description can be disabled by editing on the *Error Code Description* field (see *Internal Memory*).

Note: In the following sections the termination sequence will not be displayed for better readability, but it must be always present, otherwise the commands are not parsed.

Attention: Before sending the next command, it is necessary to wait the response of the unit to the previous one.

2.3 Communication Examples

2.3.1 Python Example

In this example it is shown how to establish a simple socket communication with the module and request its firmware version (see *VER Command*).

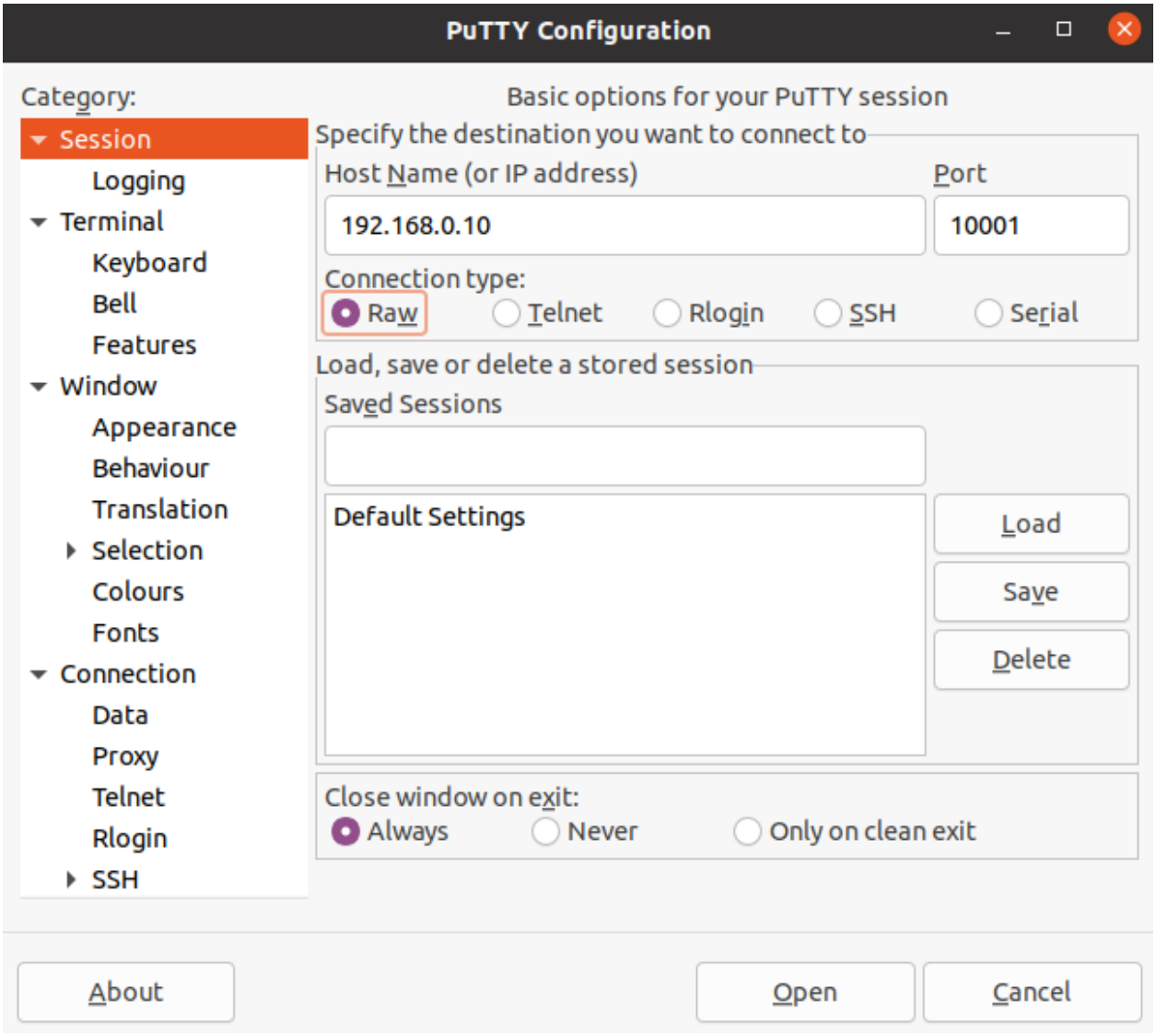
```
#!/usr/bin/env python3

import socket

IP = "192.168.0.10"

try:
    # Create TCP socket
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.connect((IP, 10001))

    # Get VERSION
    print("Get Version")
    s.sendall("VER:?\r\n".encode())
    data = s.recv(2048).decode()
    print(repr(data))

    # Close socket
    s.close()

except :
    print("Error: Error in communication with the module")
```

2.3.2 Putty Example

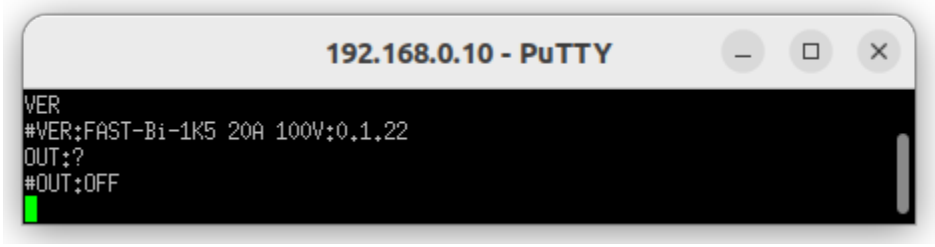
Putty it is a **free telnet client** that is available for both Windows and Linux OS. Putty allows sending/receiving commands/replies to/from the power unit.

To establish the communication socket with the unit, it is necessary to configure Putty in the following way:

- insert the IP address of the unit,
- set the communication port to **10001**
- set the connection type to *Raw*.



Once the configuration has been completed, it is possible to send the ASCII command directly to the unit:



CHAPTER

THREE

COMMANDS

In this chapter the power supply commands with its description are listed.

Note:

- Every command should be followed by the termination sequence *CRLF* `\r\n` .
- The replies from the power source are also terminated with the same sequence *CRLF* `\r\n` .

Warning: In the next examples the termination sequence will be not be shown for a better readability of the code.

3.1 Basic Commands

In this section the commands used to turn ON or OFF the power unit, change its setpoint, read the voltage, current, power etc are listed.

3.1.1 VER Command

The `VER` command returns the power supply model and the actual installed firmware version.

Command Format:

R/W	Command	Response
R	VER:?	#VER:<*name*>:<*firmware_version*>

Parameter(s):

Name	Description
<*name*>	Device name (family and model)
<*firmware_version*>	Firmware version

Example(s):

```
VER:?
#VER:BATREG2 40V 50A:1.1.03
```

3.1.2 ID Command

The `ID` command can be used to read or set the **Identification name** of the module (see *Internal Memory*) to easily identify the power supply among the other ones connected in the same network.

Command Format:

R/W	Command	Response	Description
R	ID:?	#ID:<*module_id*>	Get the module IDentification
W	ID:<*module_id*>	#AK / #NAK	Set the module IDentification

Parameters:

Name	Description
<*module_id*>	Device identification string

Tip: The default ID is the module serial number.

Tip: To change the module ID, the **ADMIN** privileges are needed (see *PASSWORD Command*)

Example(s):

```
ID:?
#ID:MAGNET_POWER_SUPPLY

ID:LAB_POWER_SUPPLY
#AK

ID:?
#ID:LAB_POWER_SUPPLY
```

3.1.3 LOOP Command

The LOOP command is used to select or get the regulation mode of the power supply.

Command Format:

R/W	Command	Response
R	LOOP:?	#LOOP:<*mode*>
W	LOOP:<*mode*>	#AK / #NAK

Parameter(s):

<*mode*>	Description
CC	*Constant Current Mode*
CV	*Constant Voltage Mode*

Tip: *Loop Mode* can be set only when the module is in the **OUT OFF State** (see *Power Supply Finite State Machine*).

Example(s):

```
LOOP:?
#LOOP:CC

LOOP:CV
#AK

LOOP:?
#LOOP:CV
```

3.1.4 UPMODE Command

The `UPMODE` command allows to select or read the setpoint *Update Mode*.

Command Format:

R/W	Command	Response
R	UPMODE:?	#UPMODE:<*mode*>
W	UPMODE:<*mode*>	#AK / #NAK

Parameter(s):

<*mode*>	Description
NORMAL	Setpoint is set using standard Ethernet communication
WAVE	Setpoint is read from the internal function generator module

Example(s):

```
UPMODE:WAVE
#AK

UPMODE:?
#UPMODE:WAVE
```

3.1.5 OUT Command

The `OUT` command can be used to enable/disable the power supply output and to monitor the status of the power supply. The power supply output state is not controlled **only** by the `OUT` command, but also by **Finite State Machine** (see *Power Supply Finite State Machine*). For this reason the `OUT` command does not control directly the power supply output, but it shifts the state of the related *FSM*, that controls the power supply output.

Command Format:

R/W	Command	Response	Notes
R	OUT:?	#OUT:<*state*>	<*state*> can be *ON* | *OFF* | *WAIT4ON*
W	OUT:<*state*>	#AK / #NAK	<*state*> can be *ON* | *OFF*

Parameter(s):

<*state*>	Mode	Description
ON	R	FSM is in **ON state**, the output relay is closed and the PID controller is running, following the selected setpoint
	W	Shifts the FSM in **Wait for ON state**
OFF	R	FSM is in **OFF state**, power stage is disabled and the PID controller is in reset
	W	Shifts the FSM in **OFF state**
WAIT4ON	R	FSM is in **Wait for ON state**, power stage is regulated to the battery voltage and the FSM evolves to *ON state*.

Example(s):

```
OUT:ON
#AK

OUT:?
#OUT:ON
```

3.1.6 SET Command

The SET command allows to set and read the current or voltage setpoint and the relative slew-rate that is used during the ramp.

Command Format:

R/W	Command	Response	Description
W	SET:I:<*sr_I*>:<*sp_I*>	#AK / #NAK	Set the current setpoint and specify the ramp slew-rate - it's applied with a **ramp**[1]
W	SET:V:<*sr_V*>:<*sp_V*>	#AK / #NAK	Set the voltage setpoint and specify the ramp slew-rate - it's applied with a **ramp**[1]
R	SET:I:?	#SET:I:<*sp_I*>	Return the current setpoint
R	SET:V:?	#SET:V:<*sp_V*>	Return the voltage setpoint
W	SET:I:<*sp_I*>	#AK / #NAK	Set the current setpoint - it is applied with a **ramp** (slew-rate in *Internal Memory* is used)
W	SET:V:<*sp_V*>	#AK / #NAK	Set the voltage setpoint - it is applied with a **ramp** (slew-rate in *Internal Memory* is used)
W	SET:I:DIRECT:?	#SET:I:DIRECT:<*sp_I*>	Return the current setpoint
W	SET:V:DIRECT:?	#SET:V:DIRECT:<*sp_V*>	Return the voltage setpoint
W	SET:I:DIRECT:<*sp_I*>	#AK / #NAK	Set the current setpoint - the setpoint is directly applied (it doesn't use ramp)
W	SET:V:DIRECT:<*sp_V*>	#AK / #NAK	Set the voltage setpoint - the setpoint is directly applied (it doesn't use ramp)
W	SET:I:RAMP:?	#SET:I:RAMP:<*sr_I*>	Return the current setpoint set at the end of the ramp
W	SET:V:RAMP:?	#SET:V:RAMP:<*sr_V*>	Return the voltage setpoint set at the end of the ramp
W	SET:I:RAMP:<*sp_I*>	#AK / #NAK	Set the current setpoint - it is applied with a **ramp** (slew-rate in *Internal Memory* is used)
W	SET:V:RAMP:<*sp_V*>	#AK / #NAK	Set the voltage setpoint - it is applied with a **ramp** (slew-rate in *Internal Memory* is used)
W	SET:I:TIME::<*time*>:<*sp_I*	#AK / #NAK	Set the current setpoint and specify the time to reach it - it's applied with a **ramp**[1]
W	SET:V:TIME::<*time*>:<*sp_*	#AK / #NAK	Set the voltage setpoint and specify the time to reach it - it's applied with a **ramp**[1]
R	SET:I:SR:?	#SET:I:SR:<*sr_I*>	Read the used current ramp slew-rate
R	SET:V:SR:?	#SET:V:SR:<*sr_V*>	Read the used voltage ramp slew-rate
W	SET:I:SR:<*sr_I*>	#AK / #NAK	Set the current ramp slew-rate
W	SET:V:SR:<*sr_V*>	#AK / #NAK	Set the voltage ramp slew-rate

Parameter(s):

Name	Type	Unit	Description
<*sp_I*>	float string	[A]	Current setpoint
<*sp_V*>	float string	[V]	Voltage setpoint
<*sr_I*>	float string	[A/s]	Current slew-rate
<*sr_V*>	float string	[V/s]	Voltage slew-rate
<*time*>	float string	[s]	Time needed to reach the setpoint

[1] the slew-rate is not updated in *Internal Memory*.

Tip: The standard set commands: `SET:I:<sp_I>` / `SET:V:<sp_V>` / `SET:I:<sr_I>:<sp_I>` / `SET:V:<sr_V>:<sp_V>` perform a **ramp** to reach the selected setpoint. The applied slew-rate is in the command or the slew-rate saved in *Internal Memory* (in case that it is not passed).

Tip: The commands `SET:I:DIRECT:<sp_I>` and `SET:V:DIRECT:<sp_V>` apply directly the setpoint without a ramp. The slope to reach the selected setpoint depends on: power supply ranges, PID parameters and load characteristics.

Note: If the system dynamics (that depends on the power supply ranges, PID parameters and load characteristics) is slower than the selected slew-rate, then the ramp will not follow exactly the slew-rate, but it will rise with the system dynamics.

Tip:

To apply a setpoint the unit has to be:

- in **ON State** (see *OUT Command*) and
- in **Normal Update mode** (see *UPMODE Command*).

Tip: To apply a **current setpoint** the *CC mode* has to be selected; vice versa to apply a **voltage setpoint** the *CV mode* has to be selected (see *LOOP Command*).

Tip: The set slew-rate should be inside the allowed range. To verify the allowed range see the *LIMITS Command*.

Warning: If the module is in some particular modes, for example in the *Trigger mode*, the current or voltage setpoint becomes active only after a **trigger event** (see *TRIGGER Command*).

Example(s):

```
SET:I:5.4
#AK

SET:I:?
#SET:I:5.4000000

SET:I:SR:?
#SET:I:SR:10.0000000

SET:I:SR:50
#AK

SET:I:SR:?
#SET:I:SR:50.0000000
```

(continues on next page)

(continued from previous page)

```
SET:I:0.5:10
#AK

SET:I:SR:?
#SET:I:SR:0.5000000
```

3.1.7 GET Command

The `GET` command allows to read the current/voltage/power value at the output of the power unit..

Command Format:

R/W	Command	Response	Description
R	GET:I:?	#GET:I:<*out_I*>	Return the output current read - *averaged value*
R	GET:V:?	#GET:V:<*out_V*>	Return the output voltage read - *averaged value*
R	GET:P:?	#GET:P:<*out_P*>	Return the output power read - *averaged value*
R	GET:I:SAMPLE:?	#GET:I:SAMPLE:<*inst_out_I*>	Return the output current read - *instantaneous value*
R	GET:V:SAMPLE:?	#GET:V:SAMPLE:<*inst_out_V*>	Return the output voltage read - *instantaneous value*
R	GET:P:SAMPLE:?	#GET:P:SAMPLE:<*inst_out_P*>	Return the output power read - *instantaneous value*
R	GET:AUX:?	#GET:AUX:<*aux_voltage*>	Return the voltage read on the auxiliary input

Parameter(s):

Name	Type	Unit	Description
<*out_I*>	float string	[A]	Output Current read (averaged)
<*out_V*>	float string	[V]	Output Voltage read (averaged)
<*out_P*>	float string	[W]	Output Power read (averaged)
<*inst_out_I*>	float string	[A]	Output Current read (instantaneous)
<*inst_out_V*>	float string	[V]	Output Voltage read (instantaneous)
<*inst_out_P*>	float string	[W]	Output Power read (instantaneous)
<*aux_voltage*>	float string	[V]	Auxiliary Voltage read

Tip: To improve the signal-to-noise ratio, the `GET:I:?` / `GET:V:?` / `GET:P:?` commands return values that are calculated using a **2-stage moving average filter**. * The first moving-average stage has a length of *8* samples, the second filter has the length of *1024* samples. * The first filter is running at a "native" frequency of is 100kHz (see *UPFREQ Command*); the second filter is receiving decimated data (by a factor *4*) from the first filter, the frequency is therefore for example: 100/4 = 25 KHz.

Tip: The **AUX Input signal is optional**, read the *Optional Boards* section to verify which extension boards mount it. The AUX measuring range is bipolar: AUX [-10V, +10V].

Example(s):

```
GET:I:?
#GET:I:5.4871231

GET:V:SAMPLE:?
#GET:V:SAMPLE:3.8769825
```

3.1.8 REG Command

The REG command allows to read the *Status Register* and the *Fault Register*, and to **reset the Fault Register**.

The *Status Register* is a *32-bit register* that contains the general information to the status of the unit such as on/off status, update mode etc. See the *Status Register* section for a detailed description of the register bits.

The *Fault Register* is a *32-bit register* that contains the fault information of the unit such (if any). See the *Fault Register* section for a detailed description of the register bits.

Command Format:

R/W	Command	Response	Description
R	REG:STATUS:?	#REG:STATUS:<*hex_value*>	Return the *Status Register* in hexadecimal representation
R	REG:FAULT:?	#REG:FAULT:<*hex_value*>	Return the *Fault Register* in hexadecimal representation[1]
R	REG:FAULT:FIRST:?	#REG:FAULT:FIRST:<*hex_value*>	Return the first fault that occurred (useful in case of multiple faults)[Page 20, 1]
W	REG:RESET	#AK / #NAK	Resets the *Fault Register*[2] and the relative *Fault bit* in the status register

Parameter(s):

Name	Type	Description
<*hex_value*>	hex string	Hex string indicating the content of the relative register

Example(s):

```
REG:STATUS:?
#REG:STATUS:0x14

REG:FAULT:?
#REG:FAULT:0x9

REG:FAULT:FIRST:?
#REG:FAULT:0x8

REG:RESET
#AK

REG:STATUS:?
#REG:STATUS:0x10
```

[1] The *Fault Register* is **latching**, so if a fault condition occurs, its information is stored in the **Fault Register** until the reset of the status register (so if, for example, a fault condition occurs and is later solved, the relative bit in the fault register remains valid, until the reset command).

[2] In order to put the module in ON (see *OUT Command*), it is necessary to solve all the fault conditions and reset the *Fault Register* before powering ON the module.

3.1.9 LIMITS Command

The `LIMITS` command can be used to query the unit's current, voltage and slew-rate limits.

Command Format:

R/W	Command	Response	Description
R	LIMITS:I:HW:?	#LIMITS:I:HW:<*minHwI*>:<*maxHwI*>	Return the **Current hardware limits** of the power unit (defined by the unit's hardware)
R	LIMITS:V:HW:?	#LIMITS:V:HW:<*minHwV*>:<*maxHwV*>	Return the **Voltage hardware limits** of the power unit (defined by the unit's hardware)
R	LIMITS:P:HW:?	#LIMITS:P:HW:<*minHwP*>:<*maxHwP*>	Return the **Power hardware limits** of the power unit (defined by the unit's hardware)
R	LIMITS:I:SW:?	#LIMITS:I:SW:<*minSwI*>:<*maxSwI*>	Return the **Current software limits** of the power unit (defined by the user)
R	LIMITS:V:SW:?	#LIMITS:V:SW:<*minSwV*>:<*maxSwV*>	Return the **Voltage software limits** of the power unit (defined by the user)
R	LIMITS:I:SR:?	#LIMITS:I:SR:<*minSrI*>:<*maxSrI*>	Return the **Current slew-rate limits** of the power unit
R	LIMITS:V:SR:?	#LIMITS:V:SR:<*minSrV*>:<*maxSrV*>	Return the **Voltage slew-rate limits** of the power unit

Parameter(s):

Name	Type	Unit	Description
<*minHwI*>	float string	[A]	Lower Current Hardware limit
<*maxHwI*>	float string	[A]	Upper Current Hardware limit
<*minHwV*>	float string	[V]	Lower Voltage Hardware limit
<*maxHwV*>	float string	[V]	Upper Voltage Hardware limit
<*minHwP*>	float string	[W]	Lower Power Hardware limit
<*maxHwP*>	float string	[W]	Upper Power Hardware limit
<*minSwI*>	float string	[A]	Lower Current Software limit
<*maxSwI*>	float string	[A]	Upper Current Software limit
<*minSwV*>	float string	[V]	Lower Voltage Software limit
<*maxSwV*>	float string	[V]	Upper Voltage Software limit
<*minSrI*>	float string	[A/s]	Lower Current Slew-rate limit
<*maxSrI*>	float string	[A/s]	Upper Current Slew-rate limit
<*minSrV*>	float string	[V/s]	Lower Voltage Slew-rate limit
<*maxSrV*>	float string	[V/s]	Upper Voltage Slew-rate limit

Example(s):

```
LIMITS:I:HW:?
#LIMITS:I:HW:-100:100

LIMITS:V:SW:?
#LIMITS:V:SW:-20.1:20.1
```

(continues on next page)

(continued from previous page)

```
LIMITS:V:SR:?
#LIMITS:V:SR:0:2000
```

3.1.10 TEMP Command

The `TEMP` command allows to read the internal and external temperatures.

Command Format:

R/W	Command	Response
R	TEMP:<*temp_id*>:?	#TEMP:<*temp_id*>:<*temp_float*>

Parameter(s):

<*temp_id*>	Description
PS	Read the temperature of the **Power Stage**
SI	Read the temperature of the **Current Sensing Shunt Resistor**
TC	Read the temperature of the **External Thermocouple** (if available - see *Optional Boards*)

Name	Type	Description
<*temp_float*>	float string	Temperature in Celsius [°C]

Warning:

- The *TEMP:TC:?* command is available only if the optional board is installed - see *Optional Boards*

Example(s):

```
TEMP:PS:?
#TEMP:PS:58.7
```

3.1.11 UPFREQ Command

The UPFREQ command returns the power supply *Update Frequency* used for the ADC readings and PID calculation.

Command Format:

R/W	Command	Response
R	UPFREQ:?	#UPFREQ:<*update_freq*>

Parameter(s):

Name	Type	Unit	Description
<*update_freq*>	float string	Hz	Update frequency for ADC readings, PID calculations and Waveform update

Tip: The ADC readings, PID calculation frequency and the PWM period, are all related and equal to the *Update Frequency*.

Tip: The update frequency is 100kHz.

Example(s):

```
UPFREQ:?
#UPFREQ:100000.00
```

3.1.12 INTERLOCK Command

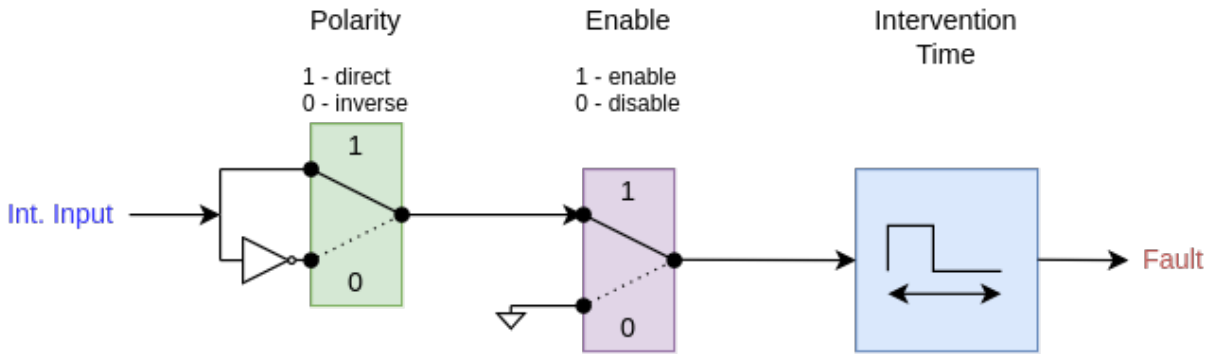
The *INTERLOCK* command allows to configure the **external interlock** inputs that are present on the unit's back panel.

In case of an external interlock trigger, the output is disabled (see the *Power Supply Finite State Machine* chapter) and the fault is reported in the *Fault Register*.

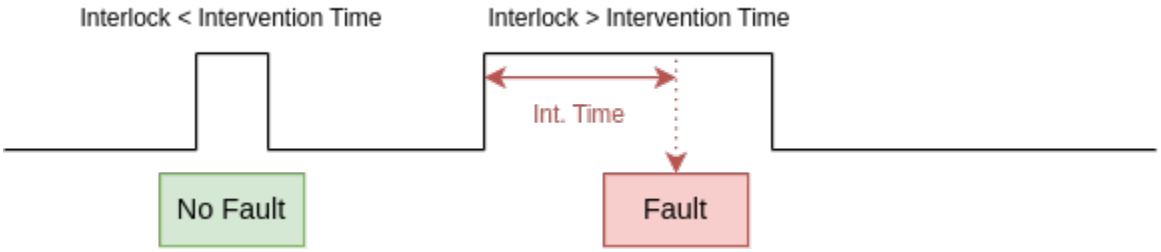
The interlock command allows to:

- enable/disable interlock detection
- set interlock polarity (direct- detect interlock when it is high; inverse - detect interlock when it is low)
- change the related interlock name (name associated to the interlock)
- set intervention time

The detecting logic for a single interlock is represented in the following figure:



The interlock intervention time specifies how much time the interlock condition should be valid in order to trip the interlock fault. It is also possible to set the intervention time to 0 -> in this case the fault trips immediately, when an interlock condition is detected.



Command Format:

R/W	Command	Response	Description
R	INTERLOCK:NUM:?	#INTERLOCK:NUM:<*int_num*>	Return the number of available interlocks on the unit
R	INTERLOCK:ENABLE:?	#INTERLOCK:ENABLE:<*enable*>	Return the interlock's enable mask
W	INTERLOCK:ENABLE:<*enable*>	#AK / #NAK	Set the interlock's enable mask
R	INTERLOCK:ENABLE:<*id*>:?	#INTERLOCK:ENABLE:<*id*>:<0|1>	Verify the enable status of the single interlock at the specified id
W	INTERLOCK:ENABLE:<*id*>:<0|1>	#AK / #NAK	Enable the the single interlock at specified id
R	INTERLOCK:POLARITY:?	#INTERLOCK:POLARITY:<*polarity*>	Return the interlock's polarity mask
W	INTERLOCK:POLARITY:<*polarity*>	#AK / #NAK	Set the interlock's polarity mask
R	INTERLOCK:POLARITY:<*id*>:?	#INTERLOCK:POLARITY:<*id*>:<0|1>	Verify the polarity status of the single interlock at specified id
W	INTERLOCK:POLARITY:<*id*>:<0|1>	#AK / #NAK	Select the polarity of the single interlock at the specified id
R	INTERLOCK:HARD:?	#INTERLOCK:HARD:<*direct*>	Return the interlock's direct mask
W	INTERLOCK:HARD:<*direct*>	#AK / #NAK	Set the interlock's direct mask
R	INTERLOCK:HARD:<*id*>:?	#INTERLOCK:HARD:<*id*>:<0|1>	Verify the hard status of the single interlock at the specified id
W	INTERLOCK:HARD:<*id*>:<0|1>	#AK / #NAK	Select the hard of the single interlock at the specified id
R	INTERLOCK:NAME:<*id*>:?	#INTERLOCK:NAME:<*id*>:<*name*>	Read the name of the interlock at the specified id
W	INTERLOCK:NAME:<*id*>:<*name*>	#AK / #NAK	Write the name of the interlock at the specified id
R	INTERLOCK:TIME:<*id*>:?	#INTERLOCK:TIME:<*id*>:<*time*>	Read the intervention time of the interlock at the specified id
W	INTERLOCK:TIME:<*id*>:<*time*>	#AK / #NAK	Write the intervention time for the interlock at the specified id

Parameter(s):

Name	Type	Description
<*int_num*>	integer string	Integer that specifies the number of available interlocks (for example "4")
<*id*>	integer string	Indicates the id of the interlock, for example 1 -> indicates interlock #1, 2 -> interlock #2 etc.
<*enable*>	hex string	Hex string that indicates the enable status for all the interlocks (1 - enabled; 0 - disabled)
<*polarity*>	hex string	Hex string that indicates the polarity status for all the interlocks (1 - direct polarity; 0 - inverse polarity)
<*direct*>	hex string	Hex string that indicates the direct status for all the interlocks (1 - hard; 0 - soft)
<*name*>	string	String associated to specified interlock
<*time*>	integer string	Intervention time, that is associated to the specified interlock. The time is specified in [ms] in the range [0,10000].

Tip: The *enable*, *polarity* and *direct* are represented by a hex mask. For example if the *enable* is 0xB -> in binary: b'1011 -> it means that interlock #4, #2 and #1 are enabled and the interlock #3 is disabled. The same for the *polarity*, so for example if the *polarity* is 0x3 -> in binary: b'11 -> it means that interlock #2 and #1 use direct polarity and the interlock #3 and #4 use inverse polarity.

Tip: The interlock name (*name*) is associated the interlock, that is visualized on the local display and on the integrated web interface in case of interlock.

See also:

Check the number of the available interlocks and the relative input circuity in the **User's Manual**.

Example(s):

```
INTERLOCK:NUM:?
#INTERLOCK:NUM:4

INTERLOCK:ENABLE:0x3
#AK

INTERLOCK:ENABLE:?
#INTERLOCK:ENABLE:0x3

INTERLOCK:ENABLE:1:0
#AK

INTERLOCK:ENABLE:1:?
#INTERLOCK:ENABLE:1:0

INTERLOCK:POLARITY:0x2
#AK

INTERLOCK:POLARITY:?
#INTERLOCK:POLARITY:0x2
```

(continues on next page)

(continued from previous page)

```
INTERLOCK:POLARITY:3:1
#AK

INTERLOCK:POLARITY:3:?
#INTERLOCK:POLARITY:3:1

INTERLOCK:NAME:2:MAGNET_INTERLOCK
#AK

INTERLOCK:HARD:0x1
#AK

INTERLOCK:HARD:?
#INTERLOCK:HARD:0x1

INTERLOCK:HARD:4:1
#AK

INTERLOCK:HARD:4:?
#INTERLOCK:HARD:4:1

INTERLOCK:NAME:2:MAGNET_INTERLOCK
#AK

INTERLOCK:NAME:2:?
#INTERLOCK:NAME:2:MAGNET_INTERLOCK

INTERLOCK:TIME:2:1000
#AK

INTERLOCK:TIME:2:?
#INTERLOCK:TIME:2:1000
```

3.2 Memory Related Commands

The following commands can be used to read and write the **Memory parameters**, that are stored in the non-volatile memory (disk). The write-access to several parameters is password protected. To modify these parameters, it is necessary to have the *admin* privileges, that can be obtained using the *PASSWORD Command* (see {ref}`PASSWORD Command`).

Certain parameters (such as calibration parameters etc.) are read-only, therefore not editable, since they are factory defined.

To read or write the memory parameters use the *MEM Command*.

The memory parameters are listed on the following page: *Internal Memory*.

3.2.1 MEM Command

The *MEM* command allows to:

- read the memory at the given memory id,
- write a value at the given memory id,
- store the memory content on the non-volatile disk.

Note: See *Internal Memory* for the whole memory description.

Command Format:

R/W	Command	Response	Description
R	MEM:<*id*>:?	#MEM:<*id*>:<*value*>	Read the memory content at the given *id*
W	MEM:<*id*>:<*value*>	#AK / #NAK	Write into the memory at the given *id*
W	MEM:SAVE	#AK	Store the memory data into the non-volatile memory

Parameter(s):

Name	Type	Range	Description
<*id*>	integer string	-[1]	*Internal Memory* field ID
<*value*>	string[1]	-[1]	Value of the *Internal Memory* field

Warning:

- After a successful writing of the memory parameter, the parameter is immediately applied, but it is not automatically stored in the non-volatile memory.
- To save the written parameter(s) in the non-volatile memory, use the *MEM:SAVE* command.

Example(s):

```
MEM:30:?
#MEM:30:TEST-UNIT

MEM:12:?
#MEM:12:1.000456

MEM:30:POWER_UNIT_ON_MAGNET_01
#AK

MEM:30:?
#MEM:30:POWER_UNIT_ON_MAGNET_01

MEM:SAVE
#AK
```

[1] see *Internal Memory*

3.2.2 PASSWORD Command

The *PASSWORD* command can be used to change the user privileges from **USER** level (default) to **ADMIN** level in order to allow the editing of the internal memory fields, which are password protected.

Tip:

- See *Internal Memory* for the whole memory description, with the associated privilege levels.
- Note: some fields are factory defined therefore not editable (read-only).

Command Format:

R/W	Command	Response	Description
R	PASSWORD:?	#PASSWORD:<*privileges*>	Get the current user's privileges level
W	PASSWORD:<*password*>	#AK / #NAK	Change the user's privileges by inserting the correct password
W	PASSWORD:NEW:<*new-psw*>	#AK / #NAK	Modify the default admin password word, with a user-defined password
W	PASSWORD:RESET:<*pin-code*>	#AK / #NAK	This command allows to restore the default admin password word

Parameter(s):

<*privileges*>	Description
USER	Lowest privileges level: it can use the unit (setting the mode of operation,enabling the output, set the current or voltage etc.), but it can only edit the basic configuration parameters
ADMIN	Highest privileges level: it can modify all available parameters and use all commands

<*password*>	Description
USER	This password can be used to exit from the *ADMIN* mode and set the *USER* mode
PS-ADMIN	*ADMIN* associated password

Tip:

- The default Admin password is **"PS-ADMIN"**
- To modify the password (command: *PASSWORD:NEW:<new-psw>*) the user has to be *ADMIN*.

Warning:

- Pay attention: if the custom password is lost it will not be possible to access with *ADMIN* privileges to the power unit.
- To restore the default password see *PASSWORD:RESET:<pin-code>* command. To get the <*pin-code*> it is necessary to **contact the CAENels support team** indicating the MAC address of the device.

- After the password reset, the default **PS-ADMIN** password is restored.
- The password is **case insensitive**

Example(s):

```
PASSWORD:?
#PASSWORD:USER

PASSWORD:PS-ADMIN
#AK

PASSWORD:?
#PASSWORD:ADMIN

PASSWORD:NEW:NEW_PASSWORD
#AK

PASSWORD:RESET:0123456789ABCDEF0123456789ABCDEF
#AK
```

3.3 Advanced Commands

In this section the advanced commands, such as trigger and waveform are described.

3.3.1 TRIGGER Command

The *TRIGGER* command allows to set and read the **trigger configuration**.

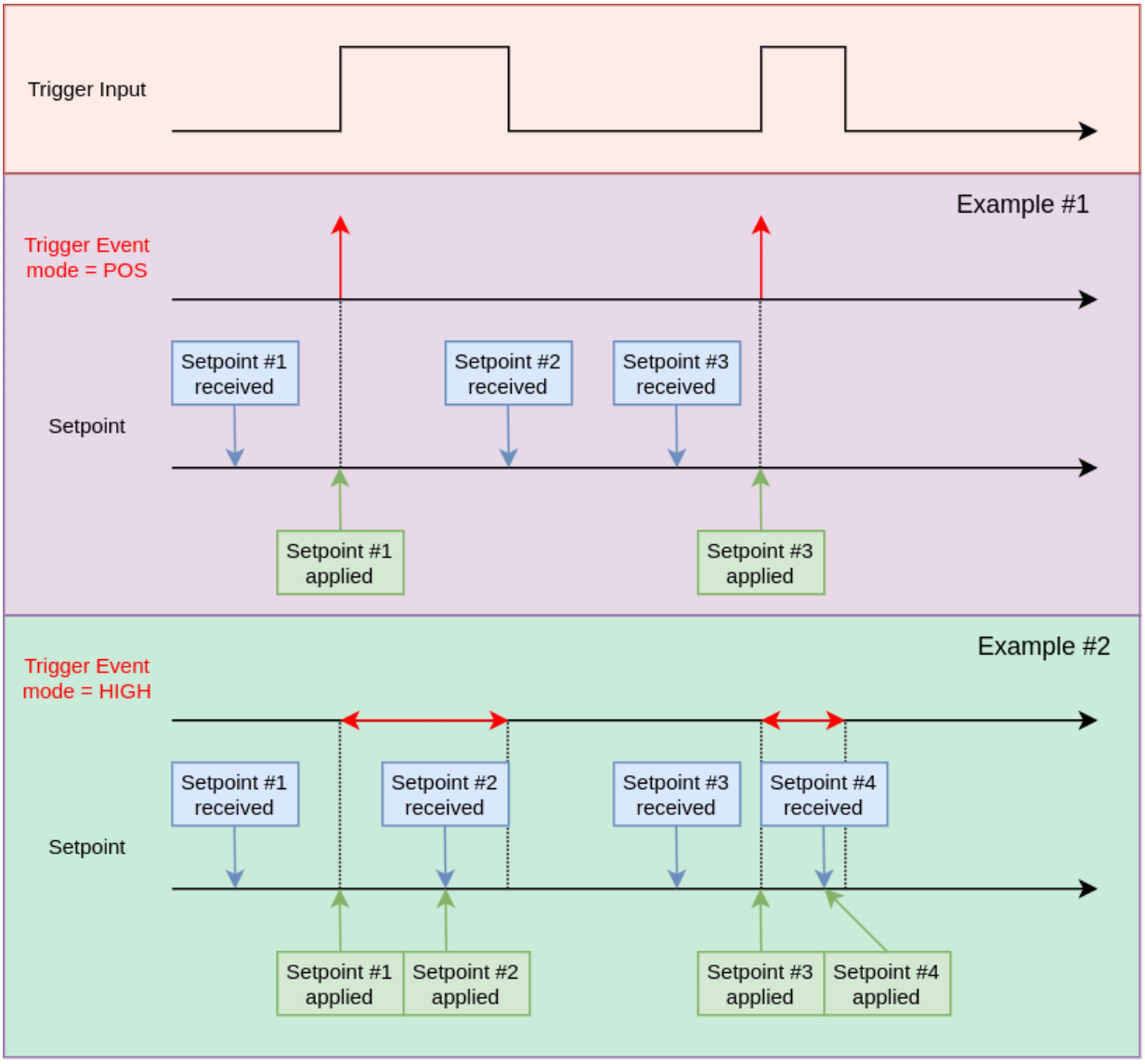
The trigger events can be configured in the following ways:

- **OFF**: Trigger input is ignored, the trigger event is not detected
- **POS**: The trigger event is detected at the **positive edge** of the trigger input signal
- **NEG**: The trigger event is detected at the **negative edge** of the trigger input signal
- **BOTH**: The trigger event is detected at **both edges (positive + negative)** of the trigger input signal
- **LOW**: The trigger event is detected when the trigger input signal **is low** (acts as a "gate signal")
- **HIGH**: The trigger event is detected when the trigger input signal **is high** (acts as a "gate signal")

The trigger functionality is also related to the **Update Mode** (see *UPMODE Command*).

When a trigger event is detected, the device performs the following actions:

Update mode	Action @ Trigger Event
NORMAL	Last setpoint or ramp is applied
WAVE	The action of the waveform depends on the Waveform Trigger configuration - see *WAVE Command*



Command Format:

R/W	Command	Response	Description
R	TRIGGER:MODE:?	#TRIGGER:MODE:<*trig_mode*>	Read the used trigger mode
W	TRIGGER:MODE:<*trig_mode*>	#AK / #NAK	Set the trigger mode
R	TRIGGER:LEVEL:?	#TRIGGER:LEVEL:<*trig_level*>	Read the detected trigger level of the trigger input signal (only for debug)

Parameter(s):

<*trig_mode*>	Description
OFF	Trigger functionality is disabled
POS	Trigger functionality is enabled - Trigger event is detected at the **positive edge** of the trigger signal
NEG	Trigger functionality is enabled - Trigger event is detected at the **negative edge** of the trigger signal
BOTH	Trigger functionality is enabled - Trigger event is detected at **both edges** of the trigger signal
LOW	Trigger functionality is enabled - Trigger event is detected when the trigger signal **is low**
HIGH	Trigger functionality is enabled - Trigger event is detected when the trigger signal **is high**

<*trig_level*>	Description
LOW	Trigger level is **low**
HIGH	Trigger level is **high**

Example(s):

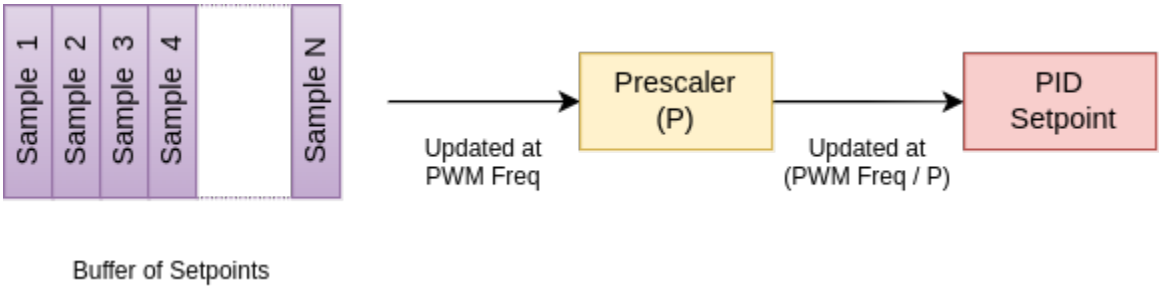
```
TRIGGER:MODE:POS
#AK

TRIGGER:MODE:?
#TRIGGER:POS

TRIGGER:LEVEL:?
#TRIGGER:LEVEL:HIGH
```

3.3.2 WAVE Command

The *WAVE* command allows to upload and reproduce a **buffer of pre-stored setpoints**. Using this command can also generate a **waveform** or reproduce a **sequence of setpoints**.

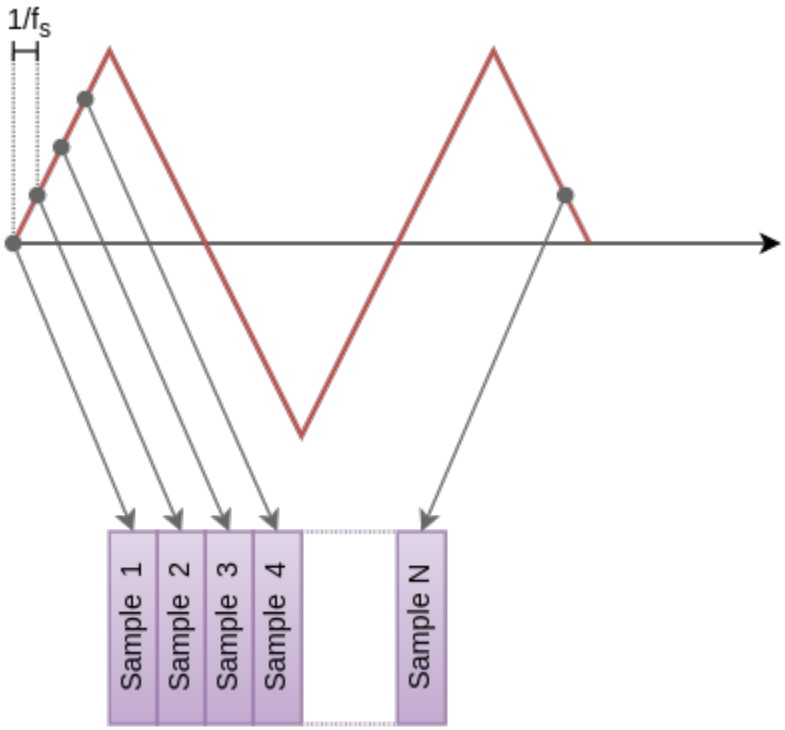
The device takes a new setpoint from the buffer, at every PID calculation cycle (*Update Frequency*, see the *UPFREQ Command*).

Since the Setpoints Buffer's length is limited, it is possible to use a **prescaler** in order to reduce the *Update Frequency* - f_{update}. This can be useful in the case of waveform in order to enlarge the waveform length or to generate very slow waveforms.



By populating the buffer with the sampled values ($f_{sampling}$) of the desired waveform it is possible to reproduce such waveform.

$$f_{sampling} = f_{update}/Prescaler$$



Command Format:

R/W	Command	Response	Description
W	WAVE:START	#AK / #NAK	Start the execution of the stored values
W	WAVE:STOP	#AK / #NAK	Stop the execution of the stored values
R	WAVE:PERIODS:?	#WAVE:PERIODS:<*period-number*>	Read the number of periods setting
W	WAVE:PERIODS:<*period-number*>	#AK / #NAK	Set the number of periods setting
R	WAVE:PERIODS:INDEX:?	#WAVE:PERIODS:INDEX:<*period-index*>	Return the currently used index of periods
R	WAVE:PRESCALER:?	#WAVE:PRESCALER:<*prescaler*>	Read the prescaler value
W	WAVE:PRESCALER:<*prescaler*>	#AK / #NAK	Set the presclaer value
R	WAVE:POINTS:NUM:?	#WAVE:POINTS:NUM:<*points-number*>	Read the number of points stored in the internal buffer
R	WAVE:POINTS:?	#WAVE:POINTS:<*p1*>:<*p2*>:...:<*pN*>	Returns the values stored in the internal buffer[1]
W	WAVE:POINTS:<*p1*>:<*p2*>:...:<*pN*>	#AK / #NAK	Fill the internal buffer with setpoints
R	WAVE:POINTS:INDEX:?	#WAVE:POINTS:INDEX:<*points-index*>	Return the currently used index of the internal buffer
R	WAVE:TRIGGER:?	#WAVE:TRIGGER:<*trigger-mode*>	Read the trigger mode
W	WAVE:TRIGGER:<*trigger-mode*>	#AK / #NAK	Set the trigger mode
W	WAVE:RAW:<*raw1*><*raw2*>...<*rawN*>	#AK / #NAK	Fill the internal buffer with setpoints (raw format)

Parameter(s):

[1] The values stored in the internal buffer may slightly differ from the set ones, because the string values, are internally converted to float numbers and so the stored number is the closest floating number to the given value.

Name	Type	Range	Description
<*period-number*>	integer string	>=0	Indicates the number of times (periods) that the buffer has to be reproduced; if equal to 0 -> Infinite periods (The waveform can be stopped with *WAVE:STOP* command)
<*period-index*>	integer string	[0, <*period-number*> - 1]	Indicates the index of the currently applied period
<*p1*>…<*pN*>	floating string	[2, 500.000]	Setpoints sequence - The min number of points is 5; the max number of setpoints is 500.000
<*points-number*>	integer string	>=0	Indicates the number of points that are stored in the buffer
<*points-index*>	integer string	[0, <*points-number*> - 1]	Indicates the index of the currently applied point
<*prescaler*>	integer string	[1, 100]	Specify the prescaler value (default value = 1)

<*trig-mode*>	Description
START	Start to reproduce the content of the buffer on trigger event or with *WAVE:START* command. To stop the execution of the waveform it is necessary to use the *WAVE:STOP* command or to wait the end of the buffer (The buffer will be reproduced <*period-number*> times).
POINTS	Reproduce the next point on edge trigger event[2] . To arm the execution it is necessary to send the *WAVE:START* command. This mode is meant to be used to reproduce a sequence of pre-defined points, **for this reason in this mode the prescaler setting is bypassed**.
GATE	In this configuration the buffer is reproduced on the trigger event[3] . The *TRIGGER* acts as an enable signal (in this mode the prescaler is **not bypassed**). To arm the waveform execution it is necessary to send the *WAVE:START* command.
GATERESET	This option has the same behavior as the *GATE* option, but when the trigger event is more active, the waveform index is reset to 0 (the waveform is rearmed).

Tip:

- The *WAVE:TRIGGER* mode is linked to the *TRIGGER* mode, please refer to {ref}`TRIGGER Command`.
- POINTS, GATE and GATERESET modes don't support TRIGGER:OFF mode.

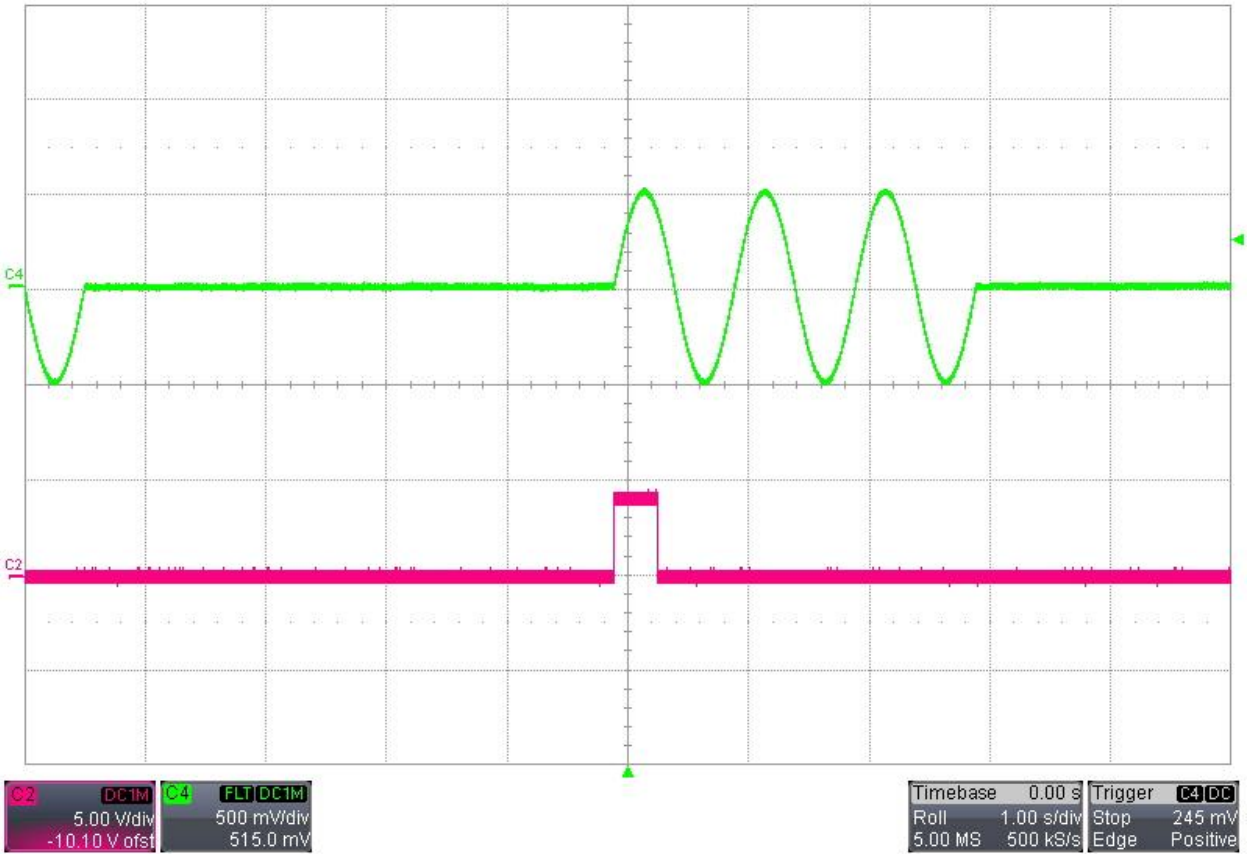
Trigger mode Examples

- For the next examples, the internal buffer was filled with points from the sine waveform with the following settings:
  - Amplitude: 1
  - Frequency: 1 Hz
  - Number of periods: 3
  - Prescaler: 1

[2] POS, NEG or BOTH *Trigger mode* are supported.
[3] HIGH or LOW *Trigger Mode* are supported.

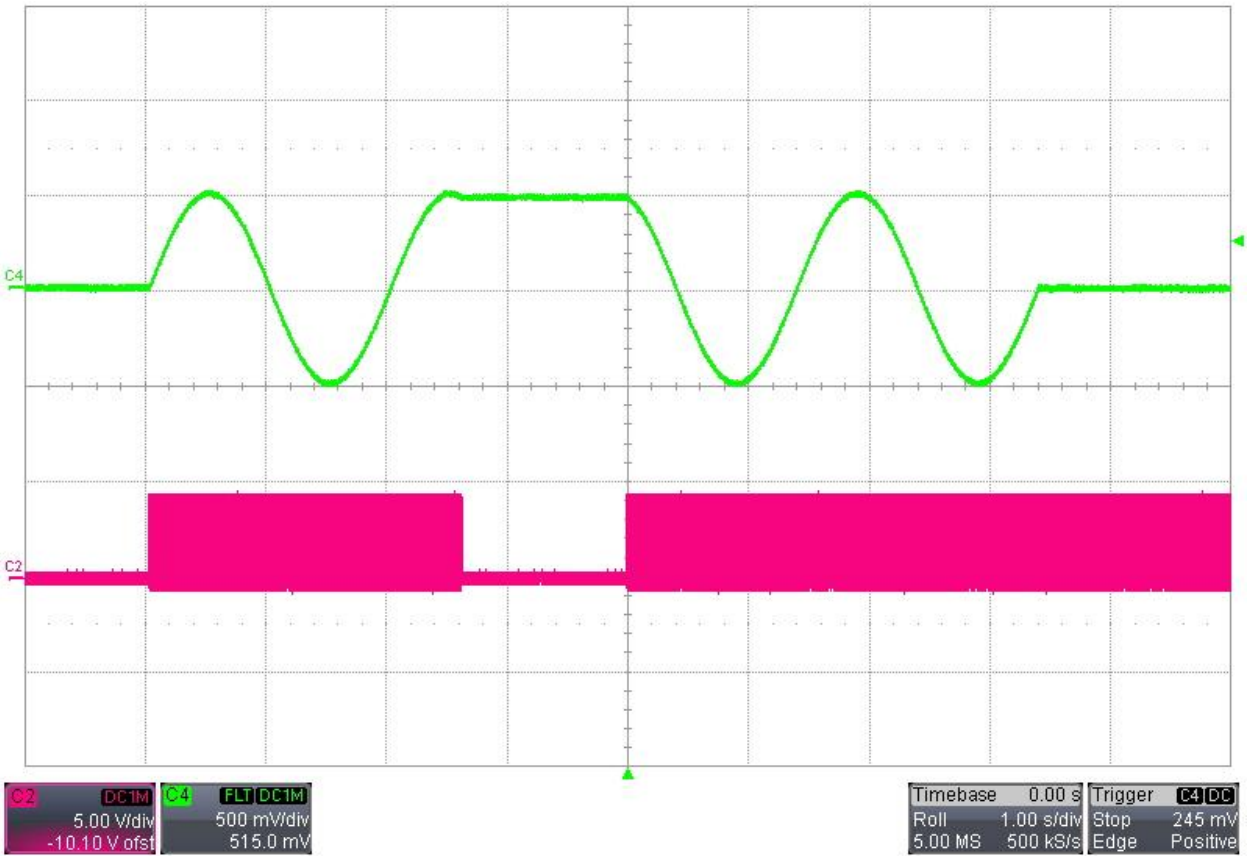
– *Example #1: WAVE:TRIGGER:START* with *TRIGGER:POS* option

In this example, the preloaded waveform **starts** when a trigger event is detected. In this example the trigger is set to *POS* and so the trigger event is generated at the *positive edge* of the trigger signal.



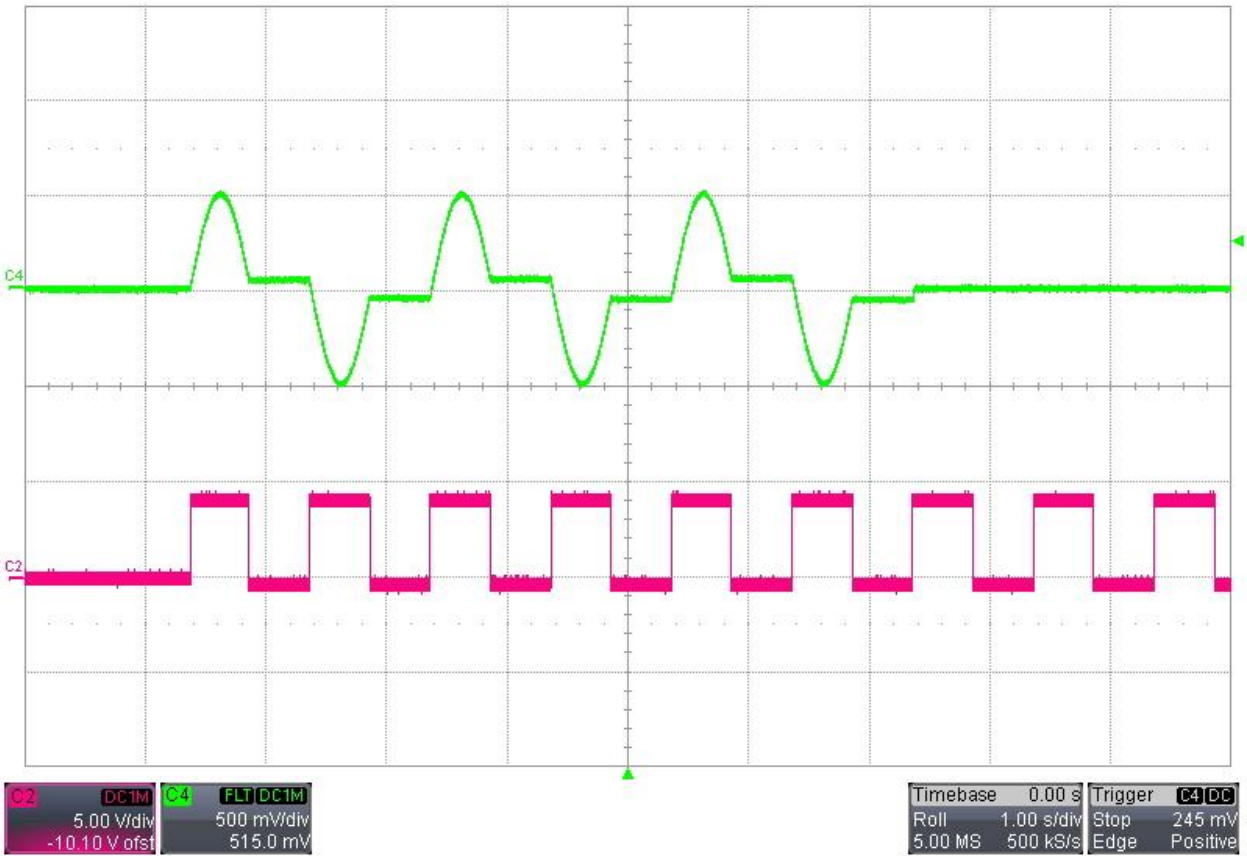
- *Example #2: WAVE:TRIGGER:POINTS* with *TRIGGER:POS* option

In this example, the preloaded waveform points are loaded when a trigger event is detected. In this example the trigger is set to *POS* and so the trigger event is generated at the *positive edge* of the trigger signal, which in this example is a square waveform at 50 KHz. When the square waveform is not executed, the trigger event is not generated and so the setpoint is not updated.



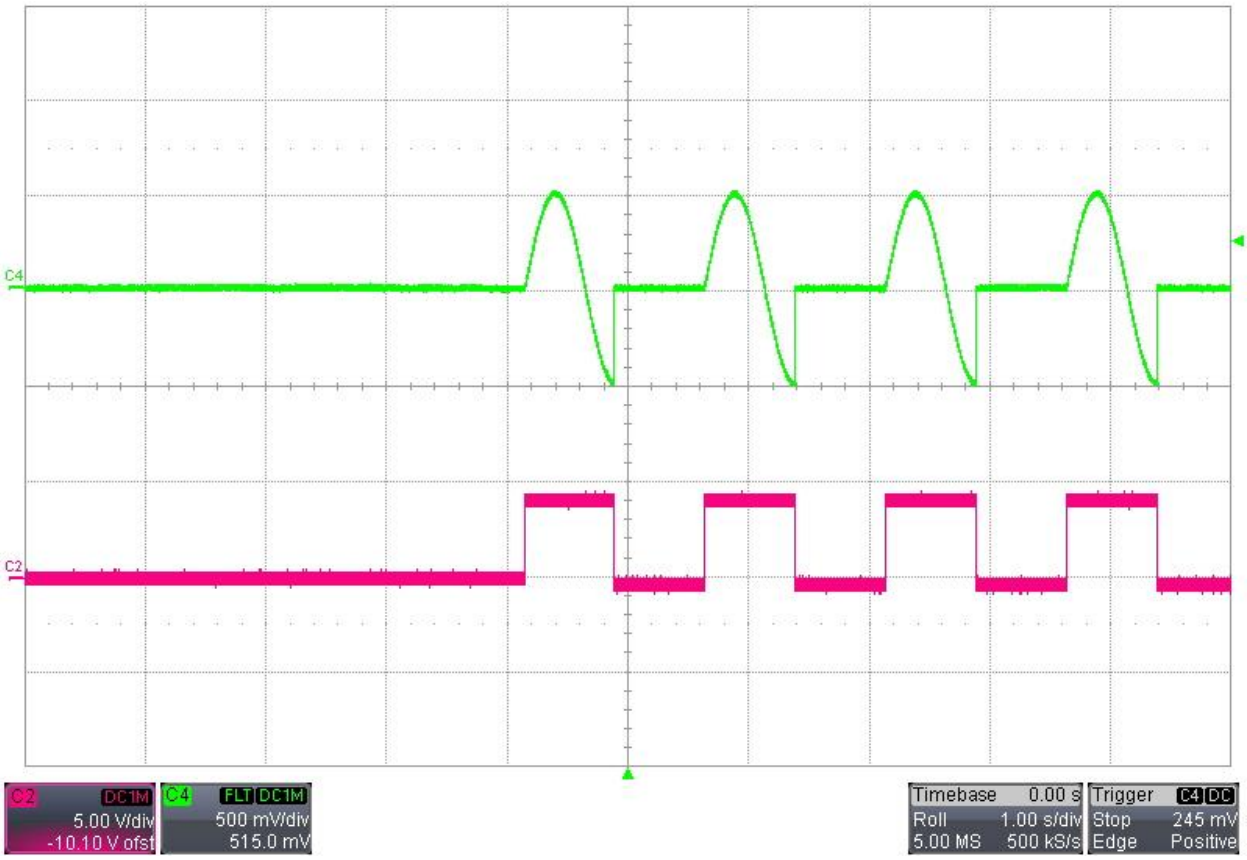
- *Example #3: WAVE:TRIGGER:GATE* with *TRIGGER:HIGH* option

In this example, the preloaded waveform points are loaded when a trigger event is detected. In this example the trigger is set to *HIGH* and so the trigger event is generated when the *trigger level is high*. In this example the trigger signal is a square waveform of 1 Hz. When the trigger signal is *high*, the trigger event is generated and the setpoint is updated.



- *Example #4: WAVE:TRIGGER:GATERESET* with *TRIGGER:HIGH* option

The configuration of this example are the same of the previous one, but the *GATERESET* mode resets the buffer index when the trigger event is not valid, and so at next trigger level, the waveform starts from the beginning.



Warning:

- The *WAVE* and *TRIGGER* commands allow also particular combinations, that does not make any sense for example:
  - When *TRIGGER:OFF* is set, the trigger sensing is disabled and so the trigger event never occurs, thus the waveform is never updated.
  - When the *WAVE:TRIGGER:GATERESET* is used with *TRIGGER:POS* configuration, the waveform is reset at every *negative edge* of the trigger -> the buffer index is constantly in reset.

Example(s):

```
WAVE:PERIODS:5
#AK

WAVE:PERIODS:?
#WAVE:PERIODS:5

WAVE:POINTS:1:2:3:4:5:6:7:8:9:10
#AK

WAVE:POINTS:NUM:?
#WAVE:POINTS:NUM:10

WAVE:PRESCALER:2
```

(continues on next page)

(continued from previous page)

```
#AK

WAVE:PRESCALER:?
#WAVE:PRESCALER:2

WAVE:TRIGGER:START
#AK

WAVE:START
#AK

WAVE:STOP
#AK
```

CHAPTER

FOUR

INTERNAL MEMORY

4.1 Internal Memory BatReg2

ID	Field	Type	Privi-leges	Description
0	Firmware ID	string	R-only	Firmware ID
1	Model	string	R-only	Model name
2	Serial Number	string	R-only	Serial Number
3	MAC Address	string	R-only	MAC Address of Ethernet
6	Hardware revision	string	R-only	Hardware revision
7	Capabilities	string	R-only	Available capabilities
8	Brand	string	R-only	Device brand
10	Module ID	string	Admin	Default value is the module s/n
12	Error Code Description	int [0|1]	Admin	Adds error description after #NAK when enabled (1)
13	Allow Remote OFF Cmd	int [0|1]	Admin	It's possible OFF the device when it's in *Local Mode*
15	TFT timeout	int	Admin	LCD timeout [minutes]
30	Current Slew Rate	float	User	Slew rate used in *CC Mode* [A/s]
31	Voltage Slew Rate	float	User	Slew rate used in *CV Mode* [V/s]
40	PID Max Voltage	float	Admin	PID Max Voltage [V]
41	PID Min Voltage	float	Admin	PID Min Voltage [V]
42	PID Max Current	float	Admin	PID Max Current [A]
43	PID Min Current	float	Admin	PID Min Current [A]
50	PID Kp_v	float	Admin	Kp parameter of Voltage PID
51	PID Ki_v	float	Admin	Ki parameter of Voltage PID
52	PID Kd_v	float	Admin	Kd parameter of Voltage PID
53	PID Kp_i	float	Admin	Kp parameter of Current PID
54	PID Ki_i	float	Admin	Ki parameter of Current PID
55	PID Kd_i	float	Admin	Kd parameter of Current PID
100	Output Over-Current Limit	float	Admin	Output over-current threshold [A]
101	Output Over-Voltage Limit	float	Admin	Output over-voltage threshold [V]
110	Max PS Temperature	float	R-only	Max Power Stage Temperature threshold [°Celsius]
112	Min TC Temperature	float	Admin	Min Thermocouple Temperature [°Celsius]
113	Max TC Temperature	float	Admin	Max Thermocouple Temperature [°Celsius]
114	TC Temperature Check Enable	int	Admin	Enable Thermocouple Temperature Check (1)
140	Interlock Enable Mask	hex	Admin	Interlocks enable mask
141	Interlock Activation Level Mask	hex	Admin	Interlocks polarity mask
142	Interlock Hard Fault Mask	hex	Admin	Interlocks direct mask
144	Interlock #1 Intervention Time	int	Admin	Interlock #1 intervention time [ms]
145	Interlock #1 Name	string	Admin	Interlock #1 name
146	Interlock #2 Intervention Time	int	Admin	Interlock #2 intervention time [ms]
147	Interlock #2 Name	string	Admin	Interlock #2 name
148	Interlock #3 Intervention Time	int	Admin	Interlock #3 intervention time [ms]
149	Interlock #3 Name	string	Admin	Interlock #3 name
150	Interlock #4 Intervention Time	int	Admin	Interlock #4 intervention time [ms]
151	Interlock #4 Name	string	Admin	Interlock #4 name
180	Calibration Date	string	R-only	Calibration date

continues on next page

Table 1 – continued from previous page

ID	Field	Type	Privi-leges	Description
181	Current Calib. Param. a	float	R-only	Current cubic calib. param. a
182	Current Calib. Param. b	float	R-only	Current cubic calib. param. b
183	Current Calib. Param. c	float	R-only	Current cubic calib. param. c
184	Current Calib. Param. d	float	R-only	Current cubic calib. param. d
185	Voltage Calib. Param. a	float	R-only	Voltage cubic calib. param. a
186	Voltage Calib. Param. b	float	R-only	Voltage cubic calib. param. b
187	Voltage Calib. Param. c	float	R-only	Voltage cubic calib. param. c
188	Voltage Calib. Param. d	float	R-only	Voltage cubic calib. param. d
191	Analog Input Calib. Param. a	float	R-only	Analog Input linear calib. param. a
192	Analog Input Calib. Param. b	float	R-only	Analog Input linear calib. param. b
193	Aux Input Calib. Param. a	float	R-only	Aux. Input linear calib. param. a
194	Aux Input Calib. Param. b	float	R-only	Aux. Input linear calib. param. b
210	Reserved	-	R-only	-
211	Reserved	-	R-only	-
212	Reserved	-	R-only	-
214	Reserved	-	R-only	-
215	Reserved	-	R-only	-
221	Acoustic Alarm Enable	int	Admin	Enable Acoustic Alarm (1)
225	DC-Link Voltage	float	R-only	DC-Link Voltage [V]

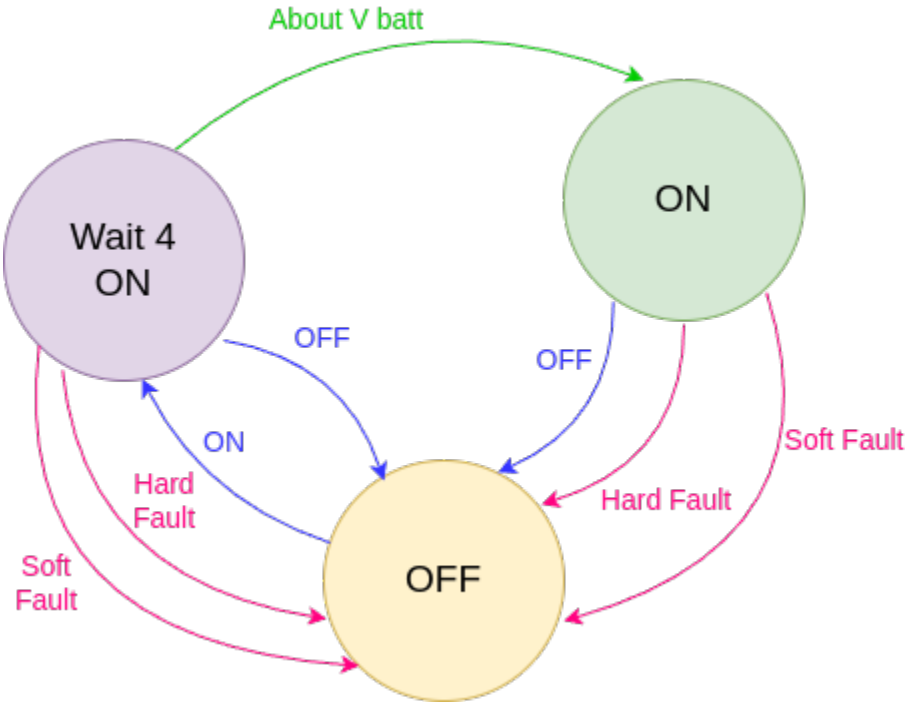
CHAPTER

FIVE

POWER SUPPLY FINITE STATE MACHINE

The behaviour of the power unit output is determinated by a **Finite State Machine (FSM)**.

The **Output Finite State Machine** is the following:



The three FSM states are:

- **OFF State**: in this state the output drivers and PID controller are disabled. This is the default state at machine startup.
- **ON State: in this state the output drivers and PID controller are enabled and the unit can accepts a new setpoint. The default setpoints are:**
  - 0A in *CC Mode*
  - Battery Voltage in *CV Mode*
- **Wait for On State**: in this state the unit sets internally the output to the battery voltage with the output disconnected. When the output reaches the battery voltage, the FSM evolves automatically to **ON State**.

The transition between the states are the following:

- **Command-related transitions** (in blue): that are defined by software commands (see *OUT Command*)

- **Fault-related transitions** (in red): that are defined by a **Fault condition**. When a fault occurs, the machine starts a transition to a "safe state", this is the **OFF State** in which the power stage is disabled.

There are two types of *Fault conditions*:

  - **Soft Fault:** faults that does NOT impact the functionality of the power unit (for example a *temperature fault*).
  - **Hard Fault:** faults that impact directly the correct functionality of the power unit (for example a *DCCT fault*).

- **Internal transitions** (in green): there is only one internal transition that is related to the output voltage. This transition is used only when the *OUT FSM* is in **OUT Wait for On State** and the output reaches the battery voltage, at this point the *OUT FSM* automatically evolves to **OUT ON State**.

CHAPTER

SIX

INTERNAL REGISTERS

In this section the meaning of all single bits of ths internal **status and fault registers** are listed.

6.1 Status Register

The **Status Register** indicates the unit's status. To read it, see the *REG Command*.

Bit(s)	Field	Description	Note
[1:0]	Module status	00: OFF	see *OUT Command*
		01: ON	
		11: Wait for On	
[2]	Fault bit	*Fault state*	see *Fault Register*
[3]	Reserved	-	
[4]	Loop mode	0: CC	see *LOOP Command*
		1: CV	
[6:5]	Reserved	-	
[7]	PID Limitation bit	PID is in saturation	The device output current/voltage is limited
[9:8]	Update mode	00: Normal	see *UPMODE Command*
		10: Waveform	
[11:10]	Reserved	-	
[12]	Local mode[1]	0: Remote	Local display commands are rejected
		1: Local	Only local display commands are accepted
[15:13]	Reserved	-	
[18:16]	Trigger mode	000: Trigger disabled	see *TRIGGER Command*
		001: Pos. Edge	
		010: Neg. Edge	
		011: Both Edges	
		100: High Level	
		101: Low Level	
[20:19]	Reserved	-	
[21]	Ramping	A *ramp* is performing	
[23]	Reserved	-	
[24]	Waveform execution	A *waveform* is performing	
[26:25]	Waveform mode	00: Start	see *WAVE Command*
		01: Point	

continues on next page

Table 1 – continued from previous page

Bit(s)	Field	Description	Note
		10: Gate	
		11: Gate reset	
[27]	Reserved	-	
[31:28]	Reserved	-	

See also:

Please see the **User's Manual** for more details.

[1] *Local/Remote mode* can be only set from the local display menu.

6.2 Fault Register

The **Fault Register** indicates the unit's fault register. To read it, see the *REG Command*.

Bit(s)	Name	Fault Type	Description
[0]	Temperature Fault	*Soft*	Over-temperature fault - see *Internal Memory id:110-111*
[1]	DC-Link Fault	*Hard*	DC-Link fault
[2]	Input OVC	*Hard*	Input Over-Current fault
[3]	Over-Power	*Soft*	Over-power fault
[7:4]	Reserved	-	-
[8]	DCCT Error	*Hard*	DCCT Error
[9]	Reserved	-	-
[10]	Output Relay Fault	*Hard*	Output Relay fault
[12:11]	Reserved	-	-
[13]	Output OVC	*Hard*	Output Over-Current Fault - see *Internal Memory id: 100*
[14]	Output OVV	*Hard*	Output Over-Voltage Fault - see *Internal Memory id: 101*
[15]	Reserved	-	-
[16]	External Interlock #1	*Soft*[1]	External Interlock #1 Fault - see *INTERLOCK Command*
[17]	External Interlock #2	*Soft*[Page 53, 1]	External Interlock #2 Fault - see *INTERLOCK Command*
[18]	External Interlock #3	*Soft*[1]	External Interlock #3 Fault - see *INTERLOCK Command*
[19]	External Interlock #4	*Soft*[1]	External Interlock #4 Fault - see *INTERLOCK Command*
[25:20]	Reserved	-	-
[26]	Thermocouple Fault	*Soft*	Thermocouple Fault - see *Internal Memory id: 112-114*
[27]	Fan(s) Fault	*Soft*	Fan(s) Fault
[31:28]	Reserved	-	-

See also:

- For a detailed description of the faults, please refer to the **User's Manual**

[1] The default *Fault Type* is set to *Soft Fault*, it's possible to change it (see *INTERLOCK Command*)

CHAPTER

SEVEN

OPTIONAL BOARDS

To extend the module's functionality it is possible to install additional boards (optional).

7.1 Available Optional Boards

Each *Optional Board* is characterized by different functionalities/associated commands.

7.1.1 FB1K5OPT0001

This board adds the following functionalities:

Option	Notes	Associated commands
Trigger Input	LVTTL 3.3V - TTL 5V tolerant	*TRIGGER Command*
Auxiliary Input	-10V, +10V	*GET Command*
K-type Thermocouple Input	-	*TEMP Command*

See also:

- For a detailed description of the faults, please refer to the **User's Manual**

CHAPTER

EIGHT

DOCUMENT REVISION

Rev	Date	Description	Firmware
1.0.4	February, 2025	Restored Internal Registers and optional boards sections	1.2.00
1.0.3	January, 2025	Added BatReg2	1.1.03
1.0.2	August, 2024	Added CROWBAR command and fixed minor errors	1.1.00
1.0.1	February, 2024	Updated internal memory	1.0.04
1.0.0	January, 2024	First documentation release in Sphinx format	1.0.00