Interface Description

English

CMD

PACEline

HBM

Hottinger Baldwin Messtechnik GmbH
Im Tiefen See 45
D-64239 Darmstadt
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com
www.hbm.com

Mat.: 7-2002.2820
DVS: A2820-2.0 en HBM: public
05.2015

1 Introduction

The digital adjustable charge amplifier CMD has an Ethernet interface for configuration and digital data-streaming and a real-time signal output as ±10 VDC voltage output.

On power-up the signal-output is -10V and both digital outputs are on logical state 0. The power-up time is 375 ms.

2 Network settings

Each unconfigured amplifier comes from production with preset unique Ethernet address and factory default settings. Default IP address is 192.168.1.10 and netmask 255.255.0.0. DHCP is enabled by default.

If there is no DHCP server on the network it can happened that development computer is not necessary on the same subnet as amplifier (IP address different from 192.168.1.X). If this is true there should be taken care about this. For this purpose IP address on computer should be set as on *Fig. 2.1*.

When the connection to amplifier is established IP address can be changed to suit desired subnet. If DHCP is enabled IP settings of amplifier are set automatically.

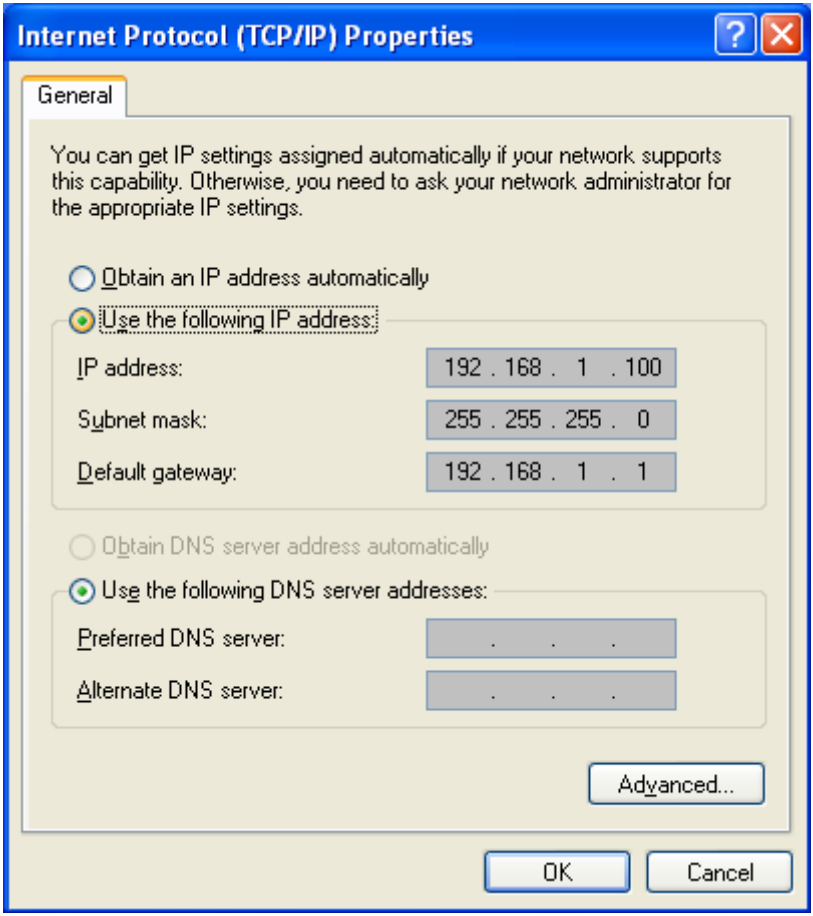


Fig. 2.1 Computer IP settings

3 Discovery procedure

When measurement application does not know the IP address of charge amplifier it can send UDP broadcast discovery packet. It should use Ethernet broadcast address (ff:ff:ff:ff:ff:ff) and IP broadcast address (ff.ff.ff.ff). The destination port should be 85 and source port should be 86. The packet should have 4 data bytes : 0x77, 0x68, 0x65, 0x72.

The whole discovery packet capture is shown on *Fig. 3.1*.

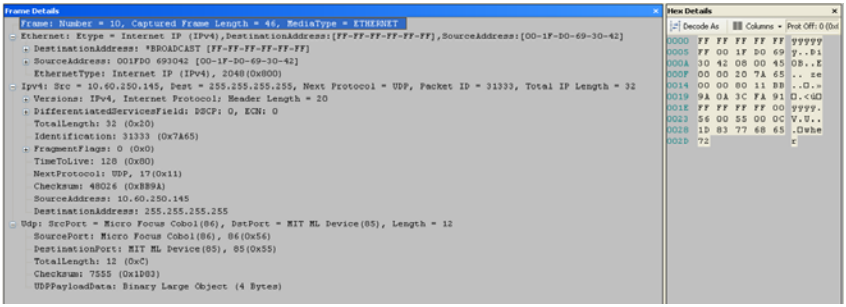


Fig. 3.1 Discovery packet capture

After reception of discovery packet all charge amplifier on the network starts to send discovery response packet. Discovery response packet is unicast UDP packet with destination IP address of discovery sender and source IP address of the amplifier. Source port is 85 and destination port is 86.

Within discovery response packet there is additional information (26 bytes) about amplifier:

- Amplifier IP
- Device ID
- Device description

Example of discovery response data definition.

```
unsigned char deviceDesc[26] =
{
                    10,60,250,143,
// textual representation of IP address (4 bytes)
                    ff, 35, a1, 0, 0, 1,
// textual representation of  amplifier ID(6 bytes)
  'E', 'm','s','i','s','o',' ','c','h','a','r','g','e','0','1',        // Device
description (15 bytes)
                    0
// end of discovery response packet
};
```

A complete discovery response packet capture is shown on

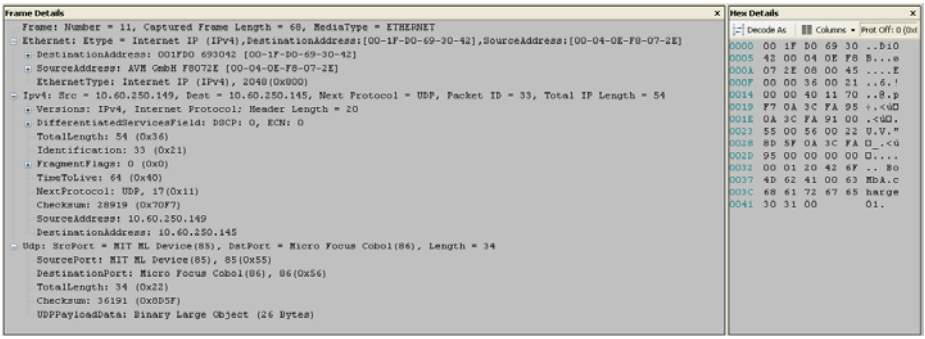


Fig. 3.2 Discovery response packet capture

After receiving discovery response packet measurement application have enough information to established telnet session with amplifier and set other parameters.

Discovery procedure II

To ensure the compatibility of CMD interface with other customer's product there is also additional discovery mechanism implemented.

To ensure the compatibility of CMD interface with other customer's product there is alsoadditional discovery mechanism implemented.

Network parameters

- Configuration mode (fixed or automatically, ie DHCP) in the case of fixed 'IP addressand subnet mask
- Router Address / Gateway Address i.e. IP address (default route): telegrams to hostswithin the connected network segments, or arising out of the configuration resultingroutes are not available, be sent to this address. An advanced configuration option formore complex routes will not be implemented because they are for the user in normal andcomplicated network topologies is not necessary
- TCP port: port number on which the component of a TCP connection in order tocommunicate with the firmware is adopted. Besides this, more ports of standard servicessuch as FTP, Telnet, ... serves
- UDP port: This adjustable receiver port number are sent real-time data

Requirements

- The configuration of all parameters of the network settings can be changed also withtextual command interface (telnet session)
- It should be possible for one or more non-configured component (s) in any network tofind and configure

- It should be possible to set up a network to scan existing components (all components)

Implementation

Parametrization

The network settings are stored in the device. Since the communication with a component on the network, will change the setting during the parameterization only after a restart effect. Restarting can also be triggered with an command.

Network scan and manual configuration

The only way to fit a non-configured network device in an IP network address, is the possibility of UDP broadcast messages to a specific port number.

The UDP layer detects valid (eg see table) telegrams using the telegram header (START: 0x12345679 The data area contains the network parameters.

Table “START”

Device Family	START
Charge amplifier	0x12345679

Table “SRV_ID”

Service	SRV_ID
Scan	0x00000001 (0)
Network configuration write	0x00000002 (1)
Network configuration read	0x00000003 (1)
Restart	0x00000004 (1)

- (0) Header in the request to send MAC address, Received -MAC = 0x000000000000
- (1) Transmit and receive MAC address in the header

Telegrammheader			
START	UINT32	0x12345679	Protokollanfang
HD_TYPE	UINT16	0x01	Header version
HD_LEN	UINT16		Header size(Telegramm-header)
MAC_SENDER	6x UINT8		Senders MAC-Adr.
MAC_RECEIVER	6x UINT8		Target MAC-Adr.
DT_LEN	UINT16		Data size
Transac-tion_Header			
TRANSTEL_TYPE	UINT16	1: Request, 2: Response	Transaction-Type
Service-Header			
SRV_ID	UINT32		Service-ID
RESULT-CODE	INT32	0 = OK, Error < 0, Warrning > 0	Error code Response, Request for 0x00000000
Data area	DT_LE N bytes		Data, optional

- Scan of a network segment is realised by sending the UDP-telegram to a broadcast address. All the modules will send answers. Explicit UDP broadcast port number = 1200 dec (may still change).

- Scan Once such a telegram was received, each module in turn, responds in the form of a UDP broadcast to the UDP broadcast port. The service data area contains data from the Network parameters: ALL strings are UTF-8 encoded and 0-terminated!

Data area

- Unit16: version of the data area for UDP responses and scanning services, »read the network configuration«: starting with 0x0001
- String: Device type
- String: Device Name
- String: firmware status
- String: Hardware status
- String: serial number
- 4 x UINT8: Tcpip Address
- 4 x UINT8: Tcpip PortNumber
- 4 x UINT8: Subnet Mask
- 4 x UINT8: Tcpip router address
- Uint16: configuration mode (fixed or DHCP = 0x0001 = 0x0002)
- 4 x UINT8: UDP port for real-time data

Binary coding of binary data: ie, Little Endian Intel

After reception of scan packets the PC(scan device) can change network parameters if they are not suitable for particular network with writing back: Service ”network configuration write”. The receiver detects correct packet with checking MAC address in header. Received settings are saved and the replay is sent:

Data Range

- version of the data range for UDP network configurations: starting with 1
- (new) address Tcpip
- (new) port number Tcpip
- (new) Subnet Mask
- (new) router Tcpip Address
- (new) configuration mode (fixed or DHCP)
- (new) UDP port for real-time data

Binary coding as before.

- To enable the new settings to take effect, the component at this point to carry out a restart, which is explicitly via service method can be triggered. Possibly. existing TCP connections will be terminated. Service: ”Restart” is + response.

4 Data Streaming

Amplifier can be used in data stream mode. When used in this mode amplifier send UDP packet with measurement results with predefine frequency.

Measurements are send in binary format (little endian) within UDP packet on predefined port. Format of message is :

voltage[V], charge[C], timestamp[ticks]

Where *timestamp* is incremented each time packet is sent, *voltage* is ADC converter value (in Volts) and *engineering unit* in measured charge (in En Unit).

The UDP packages are binary coded and comprise a header and a consecutive measured value including a time stamp.

UDP package format:

Header:

```
struct streamHeader{
unsigned char HDLen;          // Header len = 5 bytes
unsigned char HDType;       // Current header type = 0
unsigned char MType;      // Current measurement type 0
unsigned short count;      // Data running counter}
```

Consecutive measured values:

```
struct app_data {
unsigned long timestamp;
float    charge;
float    voltage;};
```

A varying number of measured values can be transmitted in an UDP package. The counter parameter in the header (counter) refers to the number of the last measured value in the package.

Provided that no package is lost during transmission, the following applies:

New count = old count + number of measured values in the package.

A time stamp is assigned to each measured value to allow easy checking of measured values.

Example (values in HEX-format):

05: Header len = 5 bytes

00: Current header type = 0

00: Current measurement type = 0

01

63: Data running counter 0x6301 = 25345dez

64

AC

26

00: unsigned long timestamp 0x0026AC64 -> 2534500dez ms

42

DC

46

C6: float charge after **IEEE 754**: 0xc646dc42 -> 12727.064

60

C6

07

C0: float voltage after **IEEE 754**: 0xc007c660 -> 2.1214828

next UDP package:

05: Header len = 5 bytes

00: Current header type = 0

00: Current measurement type = 0

02

63: Data running counter 0x6302 = 25346dez

C8

AC

26

00: unsigned long timestamp 0x0026ACC8 -> 2534600dez ms

5 Configuration via Telnet interface

Configuration is done via standard TELNET protocol. Only one TELNET session is possible in given time.

Charge amplifier accepts text commands, which are terminated with CR (hex code 0D). Optional LF characters are ignored.

Charge amplifier accepts one command at once. After command is issued, charge amplifier will respond in maximum response time. Typical response time is 10 ms without UDP streaming and 37 ms with UDP streaming (1000 values/s).

All responses from charge amplifiers begins with “OK,” or “ERROR,”.

Charge amplifiers do not distinguish between lower and uppercase characters. All input data is converted to lowercase characters before command parsing.

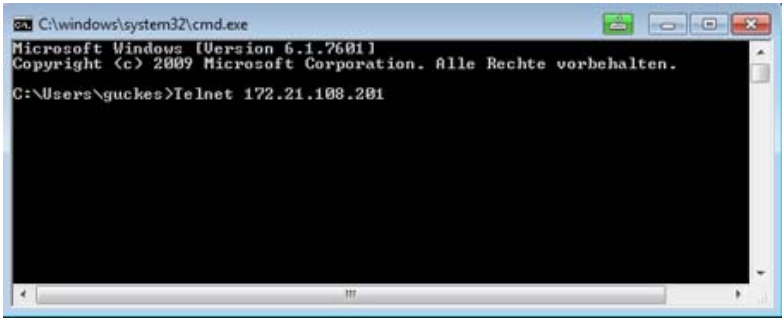
Charge amplifier default echoes back all characters. Default is restored after in each new session. This allows use of simple terminal emulators. Echo can be disabled with telnet option command. This is useful, when charge amplifier is controlled from SW application on PC.

Charge amplifier has an adjustable idle time disconnect function, see “Connection Timeout”. If there is no data in 30 minutes during a conenction, CMD sends a Live-Signal “<UNIAmp 1.0>.

5.1 Establishing the TELNET connection

1. Find IP address of CMD via the discovery procedure (Chapter 3) or via CMD Assistant -> Device-Scan for modules
2. Factory settings : IP adress 192.168.1.10 ; Sub Net mask : 255.255.0..0
3. Establish Telnet connection (Windows-Start-Button -> Run and enter: Telnet xxx.xxx.xxx.xxx)
4. Establish Telnet connection:
   a.) press windows Start-button
   b.) enter „CMD“ in command-line and start „cmd.exe“
   c.) enter: Telnet xxx.xxx.xxx.xxx

Example



5.) Enter command: e.g. factory_default

The factory settings have now been loaded. For the complete List of commands see chapter 6.

5.2 Parameters format for CMD commands

Int (integer) is signed 32 bit number.

Float numbers are always returned in float representation with 4 decimal places(example –3.4567E-9).

Decimal delimiter is period. No thousand separator is used.

6 List of commands

CH_SELECT			
Used to select active channel, which will be configured with following commands. This command adds support for multi channel charge amplifiers. In single channel amplifier is not necessary to use this command to select channel.			
command	**CH_SELECT** *channel_no*		
	parameters:		
	channel_no	int	number of channel, which will be selected. 1 is number of first channel
response	**OK, CH_SELECT** = *channel_no*		
	parameters:		
	channel_no	int	current selected channel. In single channel amplifiers this is always 1.
inquiry	**CH_SELECT** = ?		
response	**OK, CH_SELECT** = *channel_no*		
help	**CH_SELECT?**		
response	**OK, CH_SELECT *channel_no* (min = *min*_** *channel_no,* **max** = *max_ channel_no)*		
	parameters:		
	min_ channel_no	int	number of first channel. Always 1
	max_ channel_no	int	number of last channel. Always 1 in single channel versions

CH_COUNT	
Returns number of analog channels implemented in charge amplifier. In CMD this is 1.	
command	not applicable
inquiry	**CH_COUNT =** ?

response	**OK, CH_COUNT =** *nb_of_channels*
help	**CH_COUNT**?
response	**OK, CH_COUNT** *nb_of_channels*

ENGINEERING_UNIT			
Set unit on preselected channel. Unit can have up to 5 characters.			
command	**ENGINEERING_UNIT** *unit*		
	parameters:		
	unit	string	requested unit
response	**OK, ENGINEERING_UNIT** = *unit*		
	parameters:		
	unit	string	Look description above.
inquiry	**ENGINEERING_UNIT** = ?		
response	**OK, ENGINEERING_UNIT** = *unit*		
help	**ENGINEERING_UNIT** ?		
response	**OK,ENGINEERING_UNIT** *unit(max 5* characters*)*		

CH_GAIN			
Used to send gain of amplifier (for preselected channel if applicable). Gain of amplifier is dependent from selected range capacitor and PGA gain. If only PGA settings was changed, then information about charge is not lost. If also range capacitor was changed, then charge information is lost and amplifier should be reset before proceeding the measurement.			
command	**CH_GAIN** *gain*		
	parameters:		
	gain	float	requested gain in V/C
response	**OK, CH_GAIN** = *actual_gain, charge_lost, range*		
	parameters:		

	actual_gain	float	actual set gain in V/C
	Charge_lost	int	=0 … charge information preserved, range capacitor was not switched =1 … charge information lost
	range	int	=1 … only small capacitor =2 … both capacitors
inquiry	**CH_GAIN** = ?		
response	**OK, CH_GAIN** = *actual_gain, charge_lost, range*		
help	**CH_GAIN ?**		
response	**OK, CH_GAIN** gain (**min** = *min_gain,* **max** = *max_gain)*		
	parameters:		
	min_gain	float	minimum gain in V/C
	max_gain	float	maximum gain in V/C

CH_SYS_GAIN			
Used to send system gain of amplifier (for preselected channel if applicable). System gain of amplifier is dependent from ch_gain and sensor sensitivity. This gain is not stored in EEPROM. This gain actually product between sensor senditivity and channel_gain, and those two values are stored in EEPROM.			
command	**CH_SYS_GAIN** sys_*gain*		
	parameters:		
	sys_*gain*	float	requested gain in V/En_UNIT
response	**OK, CH_SYS_GAIN** = *actual_sys_gain*		
	parameters:		
	actual_sys_gain	float	actual set gain in V/En_UNIT
	charge_lost	int	=0 … charge information preserved, range capacitor was not switched =1 … charge information lost

	range	int	=1 … only small capacitor =2 … both capacitors
inquiry	**CH_SYS_GAIN** = ?		
response	**OK, CH_SYS_GAIN** = *actual_sys_gain, charge_lost, range*		
help	**CH_SYS_GAIN** ?		
response	**OK, CH_SYS_GAIN** sys_*gain* (**min** = *min_gain,* **max** = *max_gain)*		
	parameters:		
	min_gain	float	minimum gain in V/En_UNIT
	max_gain	float	maximum gain in V/En_UNIT

CH_SENSOR_SENSITIVITY			
Set sensor sensitivity of sensor attached to charge amplifier.			
command	**CH_SENSOR_SENSITIVITY** *sensitivity*		
	parameters:		
	sensitivity	float	requested *sensitivity* in C/En_Unit
response	**OK, CH_SENSOR_SENSITIVITY** = *sensitivity*		
	parameters:		
	sensitivity	float	Look description above.
inquiry	**CH_ SENSOR_SENSITIVITY** = ?		
response	**OK, CH_ SENSOR_SENSITIVITY** = *sensitivity*		
help	**CH__ SENSOR_SENSITIVITY** ?		
response	**OK, CH__ SENSOR_SENSITIVITY** *sensitivity(C/* En_Unit*)*		

CH_HPF	
Used to configure high pass filter on preselected channel	
command	**CH_HPF** *corner_ freq*

	parameters:		
	corner_ freq	float	corner frequency in Hz of high pass filter (-3dB). 0 means that high pass filter is disabled
response	**OK, CH_HPF** = *corner_ freq*		
	parameters:		
	corner_ freq	float	actual corner frequency in Hz
inquiry	**CH_HPF** = ?		
response	**OK, CH_HPF** = *corner_ freq*		
help	**CH_HPF**?		
response	**OK, CH_HPF *corner_ freq*** (*freq1, freq2, …)*		
	parameters:		
	freq1.freq_n	float	list of all possible high pass filter frequencies

CH_LPF			
Used to configure low pass filter on preselected channel.			
command	**CH_LPF** *corner_ freq*		
	parameters:		
	corner_ freq	float	corner frequency in Hz of low pass filter (-3dB).
response	**OK, CH_LPF** = *corner_ freq*		
	parameters:		
	corner_ freq	float	actual corner frequency in Hz
inquiry	**CH_LPF** = ?		
response	**OK, CH_LPF** = *corner_ freq*		
help	**CH_LPF**?		
response	**OK, CH_LPF *corner_ freq*** (*freq1, freq2, …)*		

	parameters:		
	freq1.freq_n	float	list of all possible low pass filter frequencies

CH_OUT_OFFSET			
Used to set output offset voltage for preselected channel. Example if output offset voltage is 0,5 V then charge amplifier will have 0,5 V on output if it is in reset state.			
command	**CH_OUT_OFFSET** *voltage*		
	parameters:		
	voltage	float	Offset voltage in V.
response	**OK, CH_OUT_OFFSET** = *voltage*		
	parameters:		
	voltage	float	Offset voltage in V
inquiry	**CH_OUT_OFFSET** = ?		
response	**OK, CH_OUT_OFFSET** = *voltage*		
help	**CH_OUT_OFFSET**?		
response	**OK, CH_OUT_OFFSET *voltage*** (**min** = *min_voltage,* **max** = *max_voltage)*		
	parameters:		
	min_voltage	float	Minimum offset voltage in V.
	max_voltage	float	Maximum offset voltage in V.

CH_VALUE			
Used to inquiry preselected channel value			
command	not applicable		
inquiry	**CH_VALUE**= ?		
response	**OK, CH_VALUE** = *output_voltage, output_charge, overload*		
	parameters:		
	output_voltage	float	Output value of charge amplifier in V

	output_in_EnUnit	float	Output value of charge amplifier in En_Unit
	overload	int	0.. channels was not overloaded from last reset. Default after reset of channel. 1.. channel was overloaded and charge information is lost.
help	**CH_ CH_VALUE**?		
response	**OK, CH_VALUE** *output_voltage, output_in_EnUnit, overload*		

CH_STATUS_EXTENDED			
Used to inquiry preselected channel value			
command	not applicable		
inquiry	**CH_STATUS_EXTENDED=** ?		
response	**OK, CH_STATUS_EXTENDED\r\n** *OK,output_voltage,output_in_EnUnit,overload***\r\n** *OK,min_voltage, max_voltage, min_En_Unit, max_En_Unit***\r\n** *OK,switch1_state, switch2_state***\r\n** *OK,value,channel***\r\n** *OK, reset_state, easyteach_state, digital_input_state***\r\n** *OK, output1_state,output2_state***\r\n** *OK, active_packet_set***\r\n**		
	parameters:		
	output_voltage	float	Output value of charge amplifier in V
	Output _in_EnUnit	float	Output value of charge amplifier in En_Unit
	overload	int	0.. channels was not overloaded from last reset. Default after reset of channel. 1.. channel was overloaded and charge information is lost.
	min_voltage	float	Minimum output voltage.

	max_voltage	float	Maximum output voltage.
	min_En_Unit	float	Minimum output value in En Unit.
	max_En_Unit	float	Maximum output value in En Unit.
	switch1_state	int	State of limit switch 1.
	switch2_state	int	State of limit switch 2.
	value	int	0 .. channel is in reset (its output value is 0 or equal offset voltage) 1 .. channel is operational
	ch1	int	Always 1.
	reset_state	int	Logical state of reset button input 0..button not pressed 1..button pressed
	easyteach_state	int	Logical state of easy teach button input 0..button not pressed 1..button pressed
	digital_in put _state	int	Logical state of digital input 0..button not pressed 1..button pressed
	output1_state	int	Logical state of digital output 1.
	output2_state	int	Logical state of digital output 2.
	active_packet_set	int	1..active packet set 1 2..active packet set 2
help	**CH_STATUS_EXTENDED**?		
response	**OK, CH_STATUS_EXTENDED** *extended data*		

CH_GET_PEAK_VALUES	
command	**not applicable**
inquiry	**CH_GET_PEAK_VALUES =** ?

response	**OK, CH_GET_PEAK_VALUES =** *min_voltage, max_voltage, min_En_Unit, max_En_Unit*		
	parameters:		
	min_voltage	float	Minimum output voltage.
	max_voltage	float	Maximum output voltage.
	min_En_Unit	float	Minimum output value in En Unit.
	max_En_Unit	float	Maximum output value in En Unit.
help	**CH_GET_PEAK_VALUES**?		
response	**OK, CH_GET_PEAK_VALUES** *min_voltage, max_voltage, min_En_Unit, max_En_Unit*		

CH_ANALOG_OUT_MODE			
Select possible output modes for command **CH_VALUE.** Applicable to preselected channel. Digital input in output hold mode has advantage before analog mode 1-3.			
command	**CH_ANALOG_OUT_MODE** *mode*		
	parameters:		
	mode	int	Possible mode: 0.. Measured value 1.. NegativePeak 2.. positive Peak 3.. PositivePrak-NegativePeak
response	**OK, CH_ANALOG_OUT_MODE** = *mode*		
	parameters:		
	mode	int	Look description above
inquiry	**CH_ANALOG_OUT_MODE** = ?		
response	**OK, CH_ANALOG_OUT_MODE** = *mode*		

help	**CH_ANALOG_OUT_MODE**?
response	**OK, CH_ANALOG_OUT_MODE *mode*** (0.. Measured value, 1..+Peak-(-Peak),2..-Peak, 3..+Peak ***)***

RESET_PEAK_VALUES			
Used to reset peak values for one or more channels.			
command	**RESET_PEAK_VALUES** *peak[,ch1, ch2, ..]*		
	parameters:		
	peak	int	0 .. reset max. peak value 1 .. reset min. peak value 2 .. reset both peak values
	ch1, ch2 ..	int	list of channels affected by this command. If this parameter is omitted, then command is applied to all channels.
response	**OK, RESET_PEAK_VALUES** = *peak[,channels(ch1, ch2, ..)*		
	parameters:		
	peak	int	See description above
	ch1, ch2	int	Numbers of channels, which been peak re-seted
help	**RESET_PEAK_VALUES?**		
response	**OK, RESET_PEAK_VALUES** *peak[,ch1, ch2, ..]*		

CH_LIMIT_SWITCH			
Set parameters for selected limit switch. Applicable to preselected channel.			
command	**CH_LIMIT_SWITCH** *switch, threshold[,* polarity, hysteresis]		
	parameters:		
	switch	int	1..select limit switch 1 2..select limit switch 2

	threshold	float	Limit switch value(in En_Unit)
	polarity	int	0..active if output > threshold 1.. active if output < threshold
	hysteresis	float	Only positive voltage.
response	**OK, CH_ LIMIT_SWITCH** = *switch, value,* polarity, hysteresis		
	parameters:		
	switch	int	See description above
	threshold	float	See description above
	polarity	int	See description above
	hysteresis	float	See description above
inquiry	**CH_ LIMIT_SWITCH** = ?		
response	**OK, CH_ LIMIT_SWITCH** = *1, threshold,* polarity, hysteresis		
	OK, CH_ LIMIT_SWITCH = *2, threshold,* polarity, hysteresis		
	parameters		
	threshold	float	See description above
	polarity	int	See description above
	hysteresis	float	See description above
help	**CH_ LIMIT_SWITCH** ?		
response	**OK, CH_ LIMIT_SWITCH** *switch, threshold(En_Unit),* polarity, hysteresis(En_Unit)		

CH_CLEAR_LIMIT_SWITCH			
Clear limit switch value for selected switch. Applicable to preselected channel			
command	**CH_CLEAR_LIMIT_SWITCH** *switch*		
	parameters:		
	switch	int	1..clear limit switch 1 2.. clear limit switch 2 3..clear both switches
response	**OK, CH_ CLEAR_LIMIT_SWITCH** = *switch*		

	parameters:		
	switch	int	See description above
help	**CH_ CLEAR_LIMIT_SWITCH** ?		
response	**OK, CH_ CLEAR_LIMIT_SWITCH** *switch*		

CH_GET_LIMIT_SWITCH			
Return limit switch state for selected limit switch.			
command	**CH_GET_LIMIT_SWITCH**		
response	**OK, CH_GET_LIMIT_SWITCH** = *switch1_state, switch2_state*		
	parameters:		
	switch1_ state	int	State of limit switch 1.
	switch2_ state	int	State of limit switch 2.
help	**CH_GET_LIMIT_SWITCH** ?		
response	**OK, CH_GET_LIMIT_SWITCH** *switch1_state, switch2_state*		

DIGITAL_INPUT			
Enable/disable, set polarity and mode for digital input. If mode 1 is selected then this function advantage brefore analog output mode.			
command	**DIGITAL_INPUT** *enable[, inverted,mode]*		
	parameters:		
	enable	int	0..disable digital input 1..enable digital input
	inverted	int	0..active logical state for digital input is 1 1.. active logical state for digital input is 0
	mode	int	0..clear peak values 1..hold output voltage
response	**OK, DIGITAL_INPUT** = *enable, inverted,mode*		

	parameters:		
	enable	int	See description above.
	inverted	int	See description above.
	mode	int	See description above.
inquiry	**DIGITAL_INPUT** = ?		
response	**Same as response to command**		
help	**DIGITAL_INPUT** ?		
response	**OK, DIGITAL_INPUT** *enable[, polarity,mode]*		

GET_DIGITAL_INPUT			
command	**not applicable**		
inquiry	**GET_DIGITAL_INPUT =** ?		
response	**OK, GET_DIGITAL_INPUT** = *reset_state, easyteach_state, digital_input_state*		
	parameters:		
	reset_state	int	Logical state of reset button input 0..button not pressed 1..button pressed
	easyteach_state	int	Logical state of SenorTeach button input 0..button not pressed 1..button pressed
	Digital_ in put_state	int	Logical state of digital input 0..button not pressed 1..button pressed
help	**GET_DIGITAL_INPUT** ?		
response	**OK, GET_DIGITAL_INPUT** *reset_state, easyteach_state, digital_input_state*		

SET_DIGITAL_OUTPUT			
Set/reset selected digital output. With this command can make change value of digital output only when SW controlled mode for digital output is selected.			
command	**SET_DIGITAL_OUTPUT *number, state***		
	parameters:		
	number	int	1..digital output DIG1 2..digital output DIG2
	state	int	0..reset selected digital output 1..set selected digital output
response	**OK, SET_DIGITAL_ OUTPUT =** *number , state*		
	parameters:		
	number	int	See description above.
	state	int	See description above.
inquiry	**SET_DIGITAL_OUTPUT = ?**		
response	**OK, SET_DIGITAL_OUTPUT =** *output1_state,output2_state*		
	parameters:		
	ouput1_ state	int	Logical state of digital output 1.
	ouput2_ state	int	Logical state of digital output 2.
help	**SET_DIGITAL_ OUTPUT ?**		
response	**OK, SET_DIGITAL_ OUTPUT** *number, state*		

DIGITAL_OUTPUT	
Enable/disable, set polarity and select mode for selected digital output.	
command	**DIGITAL_OUTPUT** *number*, *enable [,inverted,mode]*
	parameters:

	number	int	1..digital output DIG1 2..digital output DIG2
	enable	int	0..disable digital output 1..enable digital output
	inverted	int	0..active logical state for digital output is 1 1.. active logical state for digital output is 0
	mode	int	Possible mode: 0..state of limit switch *number* 1..overload 2..SW controlled digital output 3..System error
response	**OK, DIGITAL_ OUTPUT** = *number ,enable[, inverted, mode]*		
	parameters:		
	number	int	See description above.
	enable	int	See description above.
	inverted	int	See description above.
	mode	int	See description above.
inquiry	**DIGITAL_ OUTPUT** = ?		
response	**OK, DIGITAL_ OUTPUT**=1,*enable,inverted,mode(parameters for output 1)*		
	OK, DIGITAL_ OUTPUT=2,*enable,inverted,mode(parameters for output 2)*		
	parameters:		
	enable	int	See description above.
	inverted	int	See description above.
	mode	int	See description above.
help	**DIGITAL_ OUTPUT** ?		
response	**OK, DIGITAL_ OUTPUT** *number, enable, polarity, mode*		

CH_OVERLOAD_RESERVE			
Used to set overload reserve in auto scale mode. Applicable for preselected channel. Maximum overload reserve is 9 V.			
command	**CH_OVERLOAD_RESERVE** *voltage*		
	parameters:		
	voltage	float	Auto scale function operates so, that peak sensor charge is for ***voltage*** below max-imum amplifier output value!
response	**OK, CH_OVERLOAD_RESERVE** = *voltage*		
	parameters:		
	voltage	float	See description above
inquiry	**CH_OVERLOAD_RESERVE** = ?		
response	**OK, CH_OVERLOAD_RESERVE** = *voltage*		
	parameters:		
	voltage	float	See description above.
help	**CH_OVERLOAD_RESERVE**?		
response	**OK, CH_OVERLOAD_RESERVE** *voltage [V]*		

AUTOSCALE_ALLOW (Autoscale = SensorTeach)			
Enable/disable auto scale function.			
command	**AUTOSCALE _ALLOW** *enable*		
	parameters:		
	enable	int	0.. Disable auto scale function 1.. Enable auto scale function
response	**OK, AUTOSCALE _ALLOW =** *enable*		
	parameters:		
	enable	int	See description above
inquiry	**AUTOSCALE _ALLOW =** ?		

response	**AUTOSCALE _ALLOW =** *enable*		
	parameters:		
	enable	int	See description above.
help	**AUTOSCALE _ALLOW**?		
response	**OK, AUTOSCALE _ALLOW enable (0..disable, 1.. enable*)***		

CH_AUTOSCALE (Autoscale = SensorTeach)			
Auto scale steps for preselected channel.			
command	**CH_AUTOSCALE step**		
	parameters:		
	step	int	0…abort auto scale procedure 1… start auto scale(channel gain is set to 600nC) 2… if loaded charge between 60nC and 600nC is auto scale procedure completed. Otherwise is channel gain set to 60nC. 3… if loaded charge between 6nC and 60nC is auto scale procedure completed. Otherwise is channel gain set to 6nC. 4… if loaded charge between 500pC and 6nC is auto scale procedure completed. Otherwise is channel gain set to 500pC 5…end auto scale procedure
response	**OK, CH_ AUTOSCALE = step**		
	parameters:		
	step	int	0…auto scale procedure is aborted or auto scale procedure is completed 1…start auto scale(channel gain is set to 600nC) 2... channel gain is set to 60nC. 3...channel gain is set to 6nC. 4....channel gain is set to 500pC. 5....end of auto scale procedu

inquiry	**CH_ AUTOSCALE =** ?		
response	OK, CH_ AUTOSCALE = *step*		
	parameters:		
	step	int	0…auto scale procedure is aborted or auto scale procedure is completed 1…start auto scale(channel gain is set to 600nC) 2... channel gain is set to 60nC. 3...channel gain is set to 6nC. 4....channel gain is set to 500pC. 5....end of auto scale procedu
help	**CH_ AUTOSCALE**?		
response	**OK, CH_AUTOSCALEstep**		

EASYTEACH_ALLOW (EasyTeach = SensorTeach)			
Enable/disable easyteach function			
command	**EASYTEACH_ALLOW** *enable*		
	parameters:		
	enable	int	0.. Disable easyteach function with digital input 1.. Enable easyteach function with digital input
response	**OK, EASYTEACH_ALLOW** = *enable*		
	parameters:		
	enable	int	See description above
inquiry	**EASYTEACH_ALLOW** = ?		
response	**EASYTEACH_ALLOW** = *enable*		
help	**EASYTEACH_ALLOW**?		
response	**OK, EASYTEACH_ALLOW** enable (0..dissable, 1.. enable*)*		

CH_EASYTEACH (EasyTeach = SensorTeach)	
Start easyteach function for preselected channel	
command	**CH_EASYTEACH**
response	**OK, CH_EASYTEACH**
inquiry	**Not aplicable**
help	**CH_EASYTEACH**?
response	**OK, CH_EASYTEACH** start easyteach

RESET			
Used to reset charge in one or more channels. This command does not reset micro-processor inside charge amplifier.			
command	**RESET =** *value, [ch1, ch2, ..]*		
	parameters:		
	value	int	0 .. channel is in reset (its output value is 0 or equal offset voltage) 1 .. channel is operational
	ch1, ch2 ..	int	list of channels affected by this command. If this parameter is omitted, then command is applied to all channels.
response	**OK, RESET =** *value0 [**channels**(ch1, ch2, ..), value1 **channels**(ch3, ch4, ..)]*		
	parameters:		
	value0	int	If all channels are in the same state or this is single channel amplifier, then value0 is state of those cannels. If there are channels with different state, then value0 is 0, and then following list of all channels with this value
	ch1, ch2	int	Numbers of channels, which have state 0 (reset)
	value1	int	Always 1.

	ch3, ch4	int	list of channels, which are operational
inquiry	**RESET =** ?		
response	**Same as response to command**		
help	**RESET?**		
response	**OK, RESET *value [ch1, ch2, ..]***		

RESET_POLARITY			
Used to set polarity of reset button.			
command	**RESET_POLARITY polarity**		
	parameters:		
	polarity	int	0 .. logical low level on Reset pin is reset 1 .. logical high level on Reset pin is reset
response	**OK, RESET_ POLARITY =** *polarity*		
	parameters:		
	polarity	int	See description above.
inquiry	**RESET_ POLARITY = ?**		
response	**OK, RESET_ POLARITY = polarity**		
	parameters:		
	polarity	int	See description above.
help	**RESET_ POLARITY?**		
response	**OK, RESET_ POLARITY *value[0..logical low goes is Reset,1..log.high is Reset]***		

MANUFACTURER_DATA	
Used to inquiry data about amplifier stored during production	
command	not applicable

inquiry	**MANUFACTURER_DATA =** ?		
response	**OK, MANUFACTURER_DATA \r\n** ***manufacturer*** = *manufacturer* **\r\n*** **type** = *device_type* **\r\n*** ***firmware*** = *hardware_revision* **\r\n*** ***hardware*** = *hardware_revision* **\r\n*** ***serial*** = *serial_number* **\r\n***		
	parameters:		
	manufacturer	string	HBM
	device_type	string	CMD
	Firmware_ revision	string	Format major.minor
	Hardware_ revision	string	Format major.minor
	serial_number	string	7 digits number.
help	**MANUFACTURER_DATA**?		
response	**OK, MANUFACTURER_DATA**		

DEVICE_NAME			
Set /get user changeable device name (max 32 characters). Used to distinguish devices on LAN.			
command	**DEVICE_NAME** *name*		
	parameters:		
	name	string	Set name of charge amplifier. This string is stored in amplifier EEPROM. Unit from production has default value “New amplifier No xxxxx”, where xxxxx is amplifier serial number from label
response	**OK, DEVICE_NAME =** *name*		
	parameters:		

	name	string	See description above
inquiry	**DEVICE_NAME =** ?		
response	**OK, DEVICE_NAME =** *name*		
help	**DEVICE_NAME**?		
response	**OK, DEVICE_NAME *name* (max 32 characters*)***		

PROTECT_PARAM			
Enable/disable protection for calibration parameters. This parameter isn't changed when factory default is carried out.			
command	**PROTECT_PARAM** *enable*		
	parameters:		
	enable	int	0...parameters are not protected(all parameters can be changed) 1...parameters are protected(same parameters can't be changed)
response	**OK, PROTECT_PARAM =** *enable*		
	parameters:		
	enable	int	See description above
inquiry	**PROTECT_PARAM =** ?		
response	**OK, PROTECT_PARAM =** *enable*		
	parameters:		
	enable	int	See description above
help	**PROTECT_PARAM**?		
response	**OK, PROTECT_PARAM** *enable*		

IP_ADDRESS			
Set /get IP setting for LAN. If settings are changed, new values are active after power supply cycling or issuing command			
command	IP_ADDRESS *DHCP_mode, IP_address, telnet_port, subnet_mask, default_gateway*		
	parameters:		
	DHCP_mode	int	1.. DHCP disabled 2.. DHCP enabled(default value)
	IP_address	string	Format “XXX.XXX.XXX.XXX”. “192.168.1.1” which is default value.
	telnet_port	int	Format XXX. Default value 23.
	subnet_mode	string	Format “XXX.XXX.XXX.XXX”. Default value “255.255.255.0”
	default_gateway	string	Format “XXX.XXX.XXX.XXX”. Default value “192.168.1.0”
response	**OK, IP_ADDRESS = *DHCP_mode, IP_address, telnet_port, subnet_mode,*** *default_gateway*		
	parameters:		
	DHCP_mode	int	1.. DHCP disabled 2.. DHCP enabled(default value)
	IP_address	string	If DHCP is enabled, then returned value will be assigned IP
	telnet_port	int	See description above.
	subnet_mode	string	If DHCP is enabled, then returned value will be assigned subnet mask
	default_gateway	string	See description above.
inquiry	**IP_ADDRESS =** ?		
response	**OK, IP_ADDRESS =** *DHCP_mode, IP_address, telnet_port, subnet_mode, default_gateway*		

help	**IP_ADDRESS**?
response	**OK,IP_ADDRESS** *DHCP_mode(1..disabled,2..enabled), IP_address, telnet_port, subnet_mode, default_gateway*

CONNECTION_TIMEOUT			
If no communication with amplifier is connection abort after timeout. If setting is changed new value is active after power cycling.			
command	**CONNECTION_TIMEOUT timeout**		
	parameters:		
	timeout	int	requested timeout in seconds
response	**OK, CONNECTION_TIMEOUT = timeout**		
	parameters:		
	timeout	int	timeout in seconds
inquiry	**CONNECTION_TIMEOUT =** ?		
response	**OK, CONNECTION_TIMEOUT =** timeout in seconds		
help	**CONNECTION_TIMEOUT**?		
response	**OK, CONNECTION_TIMEOUT** timeout in seconds		

The maximum timeout can be adjusted to 524286 sec = 145 h = 6 days.

Live-Signal during a connection: every 30 s without communication CMD sends „<UNIAmp 1.0>“.

ETHERNET_ADDRESS			
Set MAC address. This address isn't changed when factory default is carried out.			
command	**ETHERNET_ADDRESS** *ethernet_address*		
	parameters:		
	Ethernet_ address	string	Format “XX:XX:XX:XX:XX:XX”
response	**OK, ETHERNET _ADDRESS =** *ethernet _address*		

	parameters:		
	Ethernet_ address	string	See description above.
inquiry	**ETHERNET _ADDRESS =** ?		
response	**OK, ETHERNET_ADDRESS =** *ethernet _address*		
	parameters:		
	Ethernet_ address	string	See description above.
help	**IP_ADDRESS**?		
response	**OK, ETHERNET _ADDRESS** *ethernet _ address*		

DATA_STREAM_TARGET			
Set /get IP address and UDP port, to which real time data will be send			
command	**DATA_STREAM_TARGET** *IP_address, UDP_port*		
	parameters:		
	IP_ address	string	Format “XXX.XXX.XXX.XXX” Default value “0.0.0.0”, which is not valid IP. Error will be returned, if data stream is enabled with this IP.
	UDP_port	int	Default value **12345.**
response	**OK, DATA_STREAM_TARGET =** *IP_address,UDP_port*		
	parameters:		
	IP_ address	string	See description above
	UDP_port	int	See description above
inquiry	**DATA_STREAM_TARGET =** ?		
response	**OK, DATA_STREAM_TARGET =** *IP_address,UDP_port*		
	IP_ address	string	See description above
	UDP_port	int	See description above

help	**DATA_STREAM_TARGET**?
response	**OK, DATA_STREAM_TARGET *IP_address, UDP_port***

DATA_STREAM_RATE			
Set /get rate at which real time data is send			
com-mand	**DATA_STREAM_RATE** *rate*		
	parameters:		
	rate	float	Rate (frequency) at which real time data is send. Minimum 1, maximum 1000. Default 1.
response	**OK, DATA_STREAM_RATE =** *actual_rate*		
	parameters:		
	actual_ rate	float	Rate (frequency) at which real time data is send. This is real value decimated to resolu-tion of internal timers in charge amplifier and is not necessary full equal to requested value
inquiry	**DATA_STREAM_RATE =** ?		
response	**OK, DATA_STREAM_RATE=** *actual_rate*		
	parameters:		
	actual_ rate	float	See description above.
help	**DATA_STREAM_RATE**?		
response	**OK, DATA_STREAM_RATE *rate (min=1Hz,max=1000Hz)***		

DATA_STREAM_ENABLED			
Enable real time measurement value transmiting to predefined IP address with pre-defined rate.			
command	**DATA_STREAM_ENABLED** *enable*		
	parameters:		
	enable	int	0.. Disable streaming 1.. Enable streaming
response	**OK, DATA_STREAM_ENABLED =** *enable*		

	parameters:		
	enable	int	See description above
inquiry	**DATA_STREAM_ENABLED =** ?		
response	**OK, DATA_STREAM_ENABLED =** *enable*		
help	**DATA_STREAM_ENABLED**?		
response	**OK, DATA_STREAM_ENABLED *enable (0.. disabled, 1.. enabled)***		

PAR_PACKET_COPY			
Used to copy source packet of parameters to destination packet of parameters.			
command	**PAR_PACKET_COPY** *source, destination*		
	parameters:		
	source	int	Source packet of parameters[1 or 2]
	destina-tion	int	Destination packet of parameters[1 or 2]
response	**OK, PAR_PACKET_COPY**		
help	**PAR_PACKET_COPY**? or =?		
response	**OK, PAR_PACKET_COPY *source_packet,destination_packet***		

PAR_PACKET_SWITCH			
Set new packet of parameters. It can be used to check which packet of parameters is active.			
command	**PAR_PACKET_SWITCH** *packet*		
	parameters:		
	packet	int	Set new packet of parameters[1 or 2]
response	**OK, PAR_PACKET_SWITCH**=*new packet*		
	parameters:		
	new packet	int	Return selected packet.

inquiry	**PAR_PACKET_SWITCH** =?		
response	**OK, PAR_PACKET_SWITCH** = *active packet*		
	parameters:		
	active packet	int	1.. active parameter packet 1. 2.. active parameter packet 2.
help	**PAR_PACKET_SWITCH**?		
response	**OK, PAR_PACKET_SWITCH** ***set new packet of parameters[1..2]***		

PAR_PACKET_POINTER

This command can be used when you want read from inactive packet. This command set read pointer to selected packet. Then is possible read parameters on packet 2
(ip_address, ch_gain, ch_lpf...). It is not recommended that you read status of device (ch_extended_status..) because values maybe won't be correct. After read must set pointer to active packet (cmd par_packet_pointer 0).

Example (Active packet is 1):

- Use command par_packet_pointer 2
- Read parameters
- use command par_packet_pointer 0

command	**PAR_PACKET_POINTER** *packet*		
	parameters:		
	packet	int	[1..2]...set read pointer to packet. 0.....set back read pointer to active packet
response	**OK, PAR_PACKET_ POINTER =** *packet*		
	parameters:		
	packet	int	Return selected packet.
inquiry	**PAR_PACKET_ POINTER =** ?		

response	**OK, PAR_PACKET_ POINTER =** *pointer*		
	parameters:		
	pointer	int	[1..2]...when read pointer is set to inactive packet 0.....pointer is set to active packet
help	**PAR_PACKET_ POINTER** ?		
response	**OK, PAR_PACKET_ POINTER *pointer to packet of parameters [0..2]***		

RESTART	
Perform full reset to charge amplifier. This is same as cycling power supply. Used to activate new network settings. Amplifier will be restarted 1 second after transmitting response to this command.	
command	**RESTART**
response	**OK, RESTART**
inquiry	**Not applicable**
help	**RESTART?**
response	**OK, RESTART used to restart amplifier**

FACTORY_DEFAULT	
This command set both packets of parameters to default values. If protection of parameters is disabled, then all parameters set to factory default, except ETHERNET_ADDRESS and protection parameter (PROTECT_PARAM). If protection of parameters is enabled, then all unprotected parameters set to factory default. This includes also IP settings. Amplifiers restore and restart will be done 1 second after transmitting response to this command. Communication with amplifier will be probably lost.	
command	**FACTORY_DEFAULT**
response	**OK, FACTORY_DEFAULT**

inquiry	**Not applicable**
help	**FACTORY_DEFAULT?**
response	**OK, FACTORY_DEFAULT restore to factory default and restart**

Factory default can be triggered also with EasyTeach button. See “Factory default procedure”.

HELP	
command	HELP
response	OK,HELP
	Print all commands with comments.

LED status

GREEN constant	Amplifier running
RED constant	Amplifier in reset
BLUE constant	Amplifier connected over TELNET session
RED blinking with 1 s period	Amplifier overloaded
BLUE blinking with 1 s period	IP not configured
RED blinking with 0,5 s period	Amplifier was powered up with easy teach line active. Boot loader is running and waiting for PC command. Time out for this state is 10 s.
WHITE	Amplifier was powered up with easy teach line active. Boot loader has finished and application is waiting 3 s. In this time factory default can be started, if reset input is pressed an released.
YELLOW blinking with 1 s period	Auto scale procedure in big measuring range.
YELLOW blinking with 0.5 s period	Auto scale procedure in small measuring range.

7 Debug commands

CH_DAC_OUT_OFFSET			
Used to set output offset voltage for preselected channel when signal from charge integrator is not connected to output stage.			
	CH_DAC_OUT_OFFSET *voltage*		
	parameters:		
	voltage	float	Offset voltage in V.
response	**OK, CH_ DAC_OUT_OFFSET =** *voltage*		
	parameters:		
	voltage	float	Offset voltage in V

help	**CH_DAC_OUT_OFFSET**?
response	**OK, CH_DAC_OUT_OFFSET** *voltage [V]*

CH_GAIN_LIMIT			
Limit the amplifier gain (ch_gain).			
command	**CH_GAIN_ LIMIT_** *select_gain,gain*		
	parameters:		
	select_gain	int	0..set minimum gain of amplifier[V/C] 1..set maximum gain of amplifier[V/C]
	gain	float	requested gain
response	**OK, CH_GAIN_ LIMIT =** *select_gain,gain*		
	parameters:		
	select_gain	int	See description above
	actual_gain	float	Actual gain of amplifier.
inquiry	**CH_GAIN_ LIMIT =** ?		
response	**OK, CH_GAIN_ LIMIT =** *min_gain, max_gain*		
	parameters:		

	min_gain	float	minimum gain of amplifier[V/C]
	max_gain	float	maximum gain of amplifier[V/C]
help	**CH_GAIN_ LIMIT**?		
response	**OK, CH_GAIN_ LIMIT** *select_gain,gain[V/C]*		

CH_INPUT_OFFSET			
Used to set input offset for first amplifier (integrator). Value is in DAC quants and corresponds to +/-3mV. This parameter isn't changed when factory default is carried out.			
command	**CH_INPUT_OFFSET** select_input_offset,*input_offset*		
	parameters:		
	select_ input_ offset	int	0...input offset, when high pass filter is off. 1...input offset for slow high pass filter. 2...input offset for fast high pass filter..
	input_offset	int	Input offset in quants.
response	**OK, CH_INPUT_OFFSET =** select_input_offset,*input_offset*		
	parameters:		
	Select_ input_ offset	int	Look decsription above.
	input_ offset	int	Input offset in quants.
inquiry	**CH_ INPUT_OFFSET =** ?		
response	**OK, CH_ INPUT_OFFSET =** *input_offset(HPF off), input_offset(slow HPF), input_offset(fast HPF)*		
help	**CH_ INPUT_OFFSET** ?		
response	**OK, CH_ INPUT_OFFSET** select_input_offset(0..2),*input_offset* (**min** = *-2048,* **max** = *2047)*		

CH_CAL_DAC
Used for DAC calibration. Sets output dac value in quants.

command	**CH_CAL_DAC** *dac_quants*		
	parameters:		
	Dac_ quants	int	DAC quants(0…4095)
response	**OK, CH_ CAL_DAC =** *dac_quants*		
	parameters:		
	Dac_ quants	int	See description above.
help	**CH_ CAL_DAC** ?		
response	**OK, CH_ CAL_DAC** *dac_quants* (**min** = *0,* **max** = *4095)*		

AD_DATA			
Return average last 50 AD value in quants. This command can not be use when streaming is enabled**.** data			
command	**Not applicable**		
command	**AD_DATA=**?		
response	**OK, AD_DATA** = *ad_quants*		
	parameters:		
	ad_quants	int	Average last 50 AD converter value in quants.
help	**AD_DATA**?		
response	**OK, AD_DATA**		

CH_PGA_GAIN			
Used to set PGA gain.			
command	**CH_PGA_GAIN** *gain*		
	parameters:		
	gain	float	requested PGA gain

response	**OK, CH_PGA_GAIN** = *actual_gain*		
	parameters:		
	actual_gain	float	actual PGA gain
inquiry	**CH_PGA_GAIN =** ?		
response	**OK, CH_PGA_GAIN =** *actual_gain*		
	parameters:		
	actual_gain	float	See description above.
help	**CH_PGA_GAIN** ?		
response	**OK, CH_PGA_GAIN gain (min =** ***1,*** **max =** ***150)***		

CH_SELECT_FILTER			
Select charge filter.			
command	**CH_SELECT_FILTER** *filter*		
	parameters:		
	filter	int	0: charge off, only DAC connected to output 1: charge on, without LPF 2: charge on, with LPF
response	**OK, CH_SELECT_FILTER =** *filter*		
	parameters:		
	filter	int	See description above.
inquiry	**CH_SELECT_FILTER =** ?		
response	**OK, CH_SELECT_FILTER =** *filter*		
help	**CH_SELECT_FILTER**?		
response	**OK, CH_SELECT_FILTER filter**		

CH_ANALOG_GAIN			
Set PGA gain with output voltage correction.			
com-mand	**CH_ANALOG_GAIN** *gain*		
	parameters:		
	gain	float	Set PGA gain
response	**OK, CH_ANALOG_GAIN =** *filter*		
	parameters:		
	actual_gain	float	Actual PGA gain
inquiry	**CH_ANALOG_GAIN =** ?		
response	**OK, CH_ANALOG_GAIN =** *actual_gain*		
	parameters:		
	actual_gain	float	See description above
help	**CH_ANALOG_GAIN**?		
response	**OK, CH_ANALOG_GAIN** *gain* (min = 1, max =100*)*		

STATS	
Send data about ethernet packets.	
command	**not applicable**
inquiry	**STATS =** ?
response	**OK,STATS \r\n**
	Data about ethernet packets.
help	**STATS**?
response	**OK, STATS**

PAR_SET			
Set parameter value for selected parameter.			
command	**PAR_SET** *nb_of_param, param_value*		
	parameters:		
	nb_of_param	int	Number of parameter.
	param_value	string	This string is stored in EEPROM.
response	**OK, PAR_SET =** *nb_of_param, param_value*		
	parameters:		
	nb_of_param	int	See description above.
	param_value	string	See description above.
help	**PAR_SET ?**		
response	**OK, PAR_SET** *nb_of_param, param_value*		

PAR_PRINT			
Print selected parameter on selected packet or print all packet of parameters.			
command	PAR_PRINT *select_packet,select_param*		
	parameters:		
	select_packet	int	Select packet.
	select_param	int	Select parameter. If is this value 255, then print all parameters in packet. In first column print name of parameter, in second column actual value of parameters and in third default value of parameters.
response	OK, PAR_PRINT = *select_param,parar_value*		
	parameters:		
	select_param	int	Selected parameter.
	parar_value	string	Parameter value for selected parameter.

inquiry	PAR_PRINT = ?		
response	OK, PAR_PRINT= actual packet		
	parameters:		
	actual packet	float	Return actual packet.
help	PAR_PRINT?		
response	OK, PAR_PRINT *select packet,select_param[255..print all packet]*		

SET_GPIO			
command	**SET_GPIO** *pin_name, pin_state*		
	parameters:		
	pin_name	string	pin_name: SW_R4, SW_R3, SW_R2, SW_C2, SW_TEST
	pin_state	int	0:set to logical 0 1:set to logical 1
response	**OK, SET_GPIO =** *pin_name, pin_state*		
	parameters:		
	pin_name	int	See description above.
	pin_state	int	See description above.

SET_RGB_LED	
Set RGB led color.	
command	**SET_RGB_COLOR** *color*
	parameters:

	color	int	0..automatic RGB control 1..Red color 2..Green color 3..Blue color
response	**OK, SET_RGB_COLOR =** *color*		
	parameters:		
	color	int	See description above.
help	**SET_RGB_COLOR**?		
response	**OK, SET_RGB_COLOR** *color (0..auto,1..red,2..green,3..blue)*		

DEBUG_HELP	
command	DEBUG_HELP
response	OK,DEBUG_HELP
	Print debug commands with comments.

8 System parameters and factory settings

CMD offers 2 parameter-sets which have the same internal structure. System parameters are the same in both parameter-sets and changes are automatically active in both.

Important

Protected Parameters are written during production and calibration and should not be changed guarantee accuracy of the CMD.

Below is a list of all parameters, which are stored in EEPROM of charge amplifier. Protected parameters are written during production and calibration. In normal use should not be accessed, because complete functionality is available via command interface.

Number	Protection	System parameter	Factory setting	Description
0	Y	Y	-300	Input offset value in quants. Used to correct input operational amplifier offset during drift compensation (min = -2048, max= 2047).
1	Y	Y	+4.9762E-3	DAC_k
2	Y	Y	-1.2002E+01	With parameters 1and 2 (DAC_n) is calculated DAC value in quants, for desired output voltage: DAC_value = (output_value-DAC_n)/DAC_k
3	Y	Y	4.3737E-4	Corr_offset(k)
4	Y	Y	-2.2000E-3	With parameters 3 and 4 Corr_offset(n) is calculated correction offset. Corr_offset = Corr_offset(k)*Actualy_PGA_gain+ Corr_offset(n).
5	Y	Y	-1.0000E-4	LPF_k
6	Y	Y	1.8000E-2	With parameters 5 and 6 (LPF_n) is calculated correction offset when LPF is enabled:
				LPF_corr_offset = LPF_k*Actualy_PGA_gain+ LPF_n
7	N	N	0.0000E+00	Offset value[V](min = 10,max=10)
8	Y	Y	4.8447E-3	AD_k

Number	Protection	System parameter	Factory setting	Description
9	Y	Y	-3.7000E-2	With parameters 8 and 9(AD_n) is calculated output voltage: Output_voltage = AD_k*AD_quants+AD_n
10	N	N	N	Engineering unit (max 5 characters).
11	N	N	1.0000E+00	Sensor sensitivity [V/EnUn]
12	Y	Y	0	Protection parameter enabled (0..protection disabled,1..protection enabled)
13	N	N	0.0000E+00	LPF frequency [Hz] (min=1,max=20000).
14	N	N	0.0000E+00	HPF frequency [Hz]. Possible frequencies: 0, 0.2, 2.
15	Y	Y	6.1841E-10	Integrator capacitor (smaller)[C/V].
16	Y	Y	6.6708E-8	Integrator capacitor (bigger) [C/V].
17	N	N	2.0000E+10	Channel gain [V/C]. Limited with parameters 17 and 18.
18	Y	Y	1.6667E+7	Min channel gain [V/C]
19	Y	Y	2.0000E+11	Max channel gain [V/C]

20	N	N	0	Select analog mode. Mode 0: measured value Mode 1: Peak to peak value Mode 2: -Peak Mode 3: +Peak
21	N	Y	0	Polarity and enable/disable of digital input.

Number	Protection	System parameter	Factory setting	Description
				0..disable input, active logical state is 1 1..disable input, active logical state is 0 2..enable input, active logical state is 1 3..enable input, active logical state is 0
22	N	Y	0	Mode for digital input (0..cleak peak values, 1..output signal hold,2..param. switch)
23	N	N	0	Digital mode for output 1. 0..state of limit switch 1..overload 2..SW controlled output 3..system error
24	N	N	0	Polarity and enable/disable of digital output 1. 0..disable output, active logical state is 1 1..disable output, active logical state is 0 2..enable output, active logical state is 1 3.. enable output, active logical state is 0
25	N	N	0	Digital mode for output 2. 0..state of limit switch 1..overload 2..SW controlled output 3..system error 4...param. switching

Number	Protection	System parameter	Factory setting	Description
26	N	N	0	Polarity and enable/disable of digital output 2. 0..disable output, active logical state is 1 1..disable output, active logical state is 0 2..enable output, active logical state is 1 3.. enable output, active logical state is 0
27	N	N	0	Limit switch 1 threshold [EnUn]
28	N	N	0	Limit switch 1 polarity. 0..active if output 1 > threshold 1.. active if output 1 < threshold
29	N	N	0	Limit switch 1 hysteresis [EnUn]
30	N	N	0	Limit switch 2 threshold [EnUn]
31	N	N	0	Limit switch 2 polarity. 0..active if output 2 > threshold 1.. active if output 2 < threshold Limit switch 2 hysteresis [EnUn]

Number	Protection	System parameter	Factory setting	Description
32	N	N	0	Limit switch 2 hysteresis [EnUn]
33	N	N	1.0000E+00	Data stream rate [Hz] (min = 1, max = 1000)
34	N	N	2	DHCP mode(1..disable, 2..enable)
35	N	N	192.168.1.1	IP address
36	N	N	23	Telnet port
37	N	N	255.255.0.0	Subnet mode
38	N	N	192.168.1.0	IP gateway
39	Y	Y	00:04:0E:F8:09:F6	Ethernet address (MAC)
40	N	N	0.0.0.0	Data stream target
41	N	N	12345	UDP port
42	N	N	1	Auto scale allow (0..disable,1..enable)
43	N	N	1.0000E+00	Overload reserve [V]
44	Y	Y	HBM	Manufacturer
45	Y	Y	CMD600 or CMD2000	Device type
46	Y	Y	0000	Serial number
47	N	N	New amplifier Nb 0000	Device name
48	N	N	60000	Connection timeout in ms
49	Y	Y	20	Input offset slow HPF
50	Y	Y	-300	Input offset fast HPF
51	Y	Y	5.0000e-4	HPF and LPF on k koef

Number	Protection	System parameter	Factory setting	Description
52	Y	Y	1.1538e-1	HPF and LPF on n koef
53	N	N	0	Reset compensation disable (0..compensation enabled, 1..comp.disabled)
54	N	N	0	Reset polarity (0..logica high level on Reset pin(on system connector) is reset, 1...logical low on Reset pin is reset)
55	N	Y	0	Address of packet of parameters.

www.hbm.com

A2820-2.0 en 7-2002.2820 HBM: public

HBM Test and Measurement
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com

measure and predict with confidence

HBM