

Low Noise / High Resolution DAC IIa

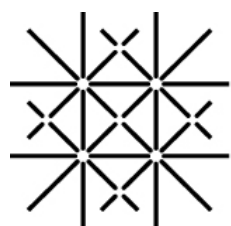
Physics Basel SP 1085

Programmer's Manual | Revision 2.1a

For Software Release 3.5.xy



Michael Steinacher | January 2026



University
of Basel

2 Table of Content

PROGRAMMER'S MANUAL REVISION 2.1A	0
2 TABLE OF CONTENT	1
1 USED ABBREVIATIONS	4
2 OVERVIEW	5
3 COMMUNICATION WITH THE LNHR DAC IIA	6
3.1 COMMUNICATION VIA SERIAL-PORT (RS-232)	6
3.2 COMMUNICATION VIA TCP/IP-TELNET PORT	8
4 GROUPING OF THE COMMANDS	11
5 SET COMMANDS (WRITE)	12
5.1 SET DAC COMMANDS	12
5.1.1 SET a DAC-Channel to a registered DAC-Value (HEX).....	12
5.1.2 SET ALL DAC-Channels to a registered DAC-Value (HEX).....	13
5.1.3 SET a DAC-Channel Status (ON or OFF)	13
5.1.4 SET ALL DAC-Channels to a Status (ON or OFF).....	13
5.1.5 SET a DAC-Channel Bandwidth (LBW=100 Hz, HBW=100 kHz)	13
5.1.6 SET ALL DAC-Channels Bandwidth (LBW=100 Hz, HBW=100 kHz)	13
5.2 SET AWG COMMANDS	14
5.2.1 SET an AWG-Memory Address (HEX) to an AWG-Value (HEX).....	14
5.2.2 SET ALL AWG-Memory Addresses to an AWG-Value (HEX).....	15
5.3 SET WAVE COMMANDS	15
5.3.1 SET a WAV-Memory Address (HEX) to a WAV-Voltage (Volt).....	16
5.3.2 SET ALL WAV-Memory Addresses to a WAV-Voltage (Volt).....	16
5.4 SET POLYNOMIAL COMMANDS	17
5.4.1 SET the POLYNOMIAL Coefficients	17
6 MULTIPLE SET COMMAND (WRITE)	18
7 QUERY COMMANDS (READ)	20
7.1 QUERY DATA COMMANDS (READ)	20
7.1.1 QUERY an Actual DAC-Value (HEX)	20
7.1.2 QUERY ALL Actual DAC-Values (HEX)	20
7.1.3 QUERY a Registered DAC-Value (HEX).....	20
7.1.4 QUERY ALL Registered DAC-Values (HEX).....	21
7.1.5 QUERY a DAC-Status (ON or OFF).....	21
7.1.6 QUERY ALL DAC-Status (ON or OFF).....	21
7.1.7 QUERY a DAC-Channel Bandwidth (LBW=100 Hz or HBW=100 kHz)	21
7.1.8 QUERY ALL DAC-Channel Bandwidths (LBW=100 Hz or HBW=100 kHz)	22
7.1.9 QUERY a DAC-MODE (ERR/DAC/SYN/RMP/AWG/ ---)	22
7.1.10 QUERY ALL DAC-MODEs (ERR/DAC/SYN/RMP/AWG/ ---)	22
7.1.11 QUERY an AWG-Value (HEX) at AWG-Address (HEX)	23
7.1.12 QUERY a BLOCK (BLK) of 1'000 (decimal) AWG-Values (HEX)	23
7.1.13 QUERY a WAV-Value (Voltage) at WAV-Address (HEX).....	23
7.1.14 QUERY a BLOCK (BLK) of 1'000 (decimal) WAV-Values (Voltage)	24
7.1.15 QUERY the POLYNOMIAL Coefficients	24
7.2 QUERY INFORMATION COMMANDS (READ)	25
7.2.1 Query "?".....	25
7.2.2 Query "HELP?".....	25
7.2.3 Query "SOFT?".....	26
7.2.4 Query "HARD?"	26
7.2.5 Query "IDN?"	26
7.2.6 Query "HEALTH?"	26
7.2.7 Query "IP?".....	26

7.2.8	Query "SERIAL?"	26
7.2.9	Query "CONTACT?"	27
8	CONTROL COMMANDS (READ / WRITE)	28
8.1	DAC UPDATE-MODE & SYNCHRONIZATION CONTROL COMMANDS	29
8.1.1	DAC Update-Mode (Read / Write)	29
8.1.2	Synchronous DAC-Update (Write only)	29
8.2	RAMP/STEP-GENERATOR CONTROL COMMANDS	30
8.2.1	RAMP Start/Hold/Stop (Write only)	30
8.2.2	RAMP State (Read only)	30
8.2.3	RAMP Cycles-Done (Read only)	30
8.2.4	RAMP Steps-Done (Read only)	31
8.2.5	RAMP Step-Size Voltage (Read only)	31
8.2.6	RAMP Steps per Cycle (Read only)	31
8.2.7	RAMP DAC-Channel AVAILABLE (Read only)	32
8.2.8	RAMP Selected DAC-CHANNEL (Read / Write)	32
8.2.9	RAMP Start Voltage (Read / Write)	32
8.2.10	RAMP Stop/Peak Voltage (Read / Write)	33
8.2.11	RAMP Time (Read / Write)	33
8.2.12	RAMP Shape (Read / Write)	34
8.2.13	RAMP Cycles-Set (Read / Write)	34
8.2.14	RAMP/STEP-Selection (Read / Write)	34
8.3	2D-SCAN CONTROL COMMANDS	36
8.3.1	Normal-Start / Auto-Start AWG (Read / Write)	36
8.3.2	Keep / Reload AWG MEM (Read / Write)	36
8.3.3	Skip / Apply Polynomial (Read / Write)	37
8.3.4	Adaptive Shift-Voltage (Read / Write)	38
8.4	AWG CONTROL COMMANDS	39
8.4.1	AWG Normal / AWG Only (Read / Write)	39
8.4.2	AWG Start / Stop (Write only)	39
8.4.3	AWG State (Read only)	40
8.4.4	AWG Cycles-Done (Read only)	40
8.4.5	AWG Duration/Period (Read only)	40
8.4.6	AWG DAC-Channel AVAILABLE (Read only)	41
8.4.7	AWG Selected DAC-CHANNEL (Read / Write)	41
8.4.8	AWG-Memory Size (Read / Write)	41
8.4.9	AWG Cycles-Set (Read / Write)	42
8.4.10	AWG External Trigger Mode (Read / Write)	42
8.4.11	AWG Clock-Period [μ sec] (Read / Write)	43
8.4.12	AWG 1 MHz Clock Reference ON/OFF (Read / Write)	44
8.5	STANDARD WAVEFORM GENERATION (SWG) CONTROL COMMANDS	45
8.5.1	SWG Mode (Read / Write)	45
8.5.2	Standard Waveform Function (Read / Write)	45
8.5.3	Desired AWG-Frequency [Hz] (Read / Write)	46
8.5.4	Keep / Adapt AWG Clock-Period (Read / Write)	46
8.5.5	Amplitude [Vp] (Read / Write)	47
8.5.6	DC-Offset Voltage [V] (Read / Write)	47
8.5.7	Phase [$^{\circ}$] (Read / Write)	48
8.5.8	Duty-Cycle [%] (Read / Write)	48
8.5.9	Wave-Memory Size (Read only)	49
8.5.10	Nearest AWG-Frequency [Hz] (Read only)	49
8.5.11	Waveform Clipping Status (Read only)	50
8.5.12	SWG/AWG Clock-Period [μ sec] (Read only)	50
8.5.13	Selected Wave-Memory (Read / Write)	50
8.5.14	Selected Wave-Function (Read / Write)	51
8.5.15	No Linearization / Linearization for actual DAC-Channel (Read / Write)	52
8.5.16	Apply Wave-Function to Wave-Memory Now (Write only)	52
8.6	WAVE CONTROL COMMANDS	53
8.6.1	Wave-Memory Size (Read only)	53

8.6.2	<i>Wave-Memory Clear (Write only)</i>	53
8.6.3	<i>Wave-Memory Save (Write only)</i>	53
8.6.4	<i>Wave-Memory Linearization DAC-Channel (Read only)</i>	53
8.6.5	<i>Write Wave-Memory to AWG-Memory (Write only)</i>	54
8.6.6	<i>Wave-Memory Writing Busy (Read only)</i>	54
9	CONVERTING DAC-VOLTAGE TO DAC-VALUE	55
10	COMPILATION OF ALL COMMANDS WITH “EXAMPLES”	56

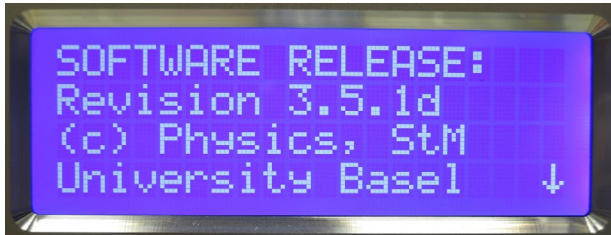
1 Used Abbreviations

The following abbreviations are used in this manual:

Abbreviation	Meaning
μsec	micro-second (1E-6)
2D	Two-Dimensional
AGND	Analog Ground
ASCII	American Standard Code for Information Interchange
AWG	Arbitrary Waveform Generator
CLK	Clock
CPU	Central Processing Unit
DAC	Digital/Analog Converter
DAQ	Data Acquisition
Float	Floating-point number
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HBW	High Bandwidth (100 kHz)
HEX	Hexadecimal
Inf	Infinite
LAN	Local Area Network
LBW	Low Bandwidth (100 Hz)
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LNHR	Low Noise / High Resolution
msec	milli-second (1E-3)
NaN	Not a Number
NIC	Network Interface Card
PC	Personal Computer
POLY	Polynomial
RMP	Ramp
RMS	Root Mean Square (Effective Value)
SWG	Standard Waveform Generation
TTL	Transistor-Transistor Logic
WAV	Wave
WF	Waveform

2 Overview

This documentation applies to the LNHR DAC Ila (SP 1085) with 24 DAC-Channels and with a Software Release of 3.5.xy (x is a number from 0 to 9 and y a character from “a” to “z”). You can easily check the installed release on the local LCD under the menu item “Software Release”:



The LNHR DAC Ila is an upgraded version of the LNHR DAC II. The new Ila version is based on NI's sbRIO-9608 FPGA/RT computer board, while the old version was based on the obsolete sbRIO-9607 FPGA/RT computer board. While most specifications remain unchanged, the new Ila version features up to six (6) AWGs, each with 65'000 points. Additionally, each AWG channel produces a point-by-point clock signal, enabling the triggering of an external DAQ at each AWG point.

Two different version of the LNHR DAC Ila are available:

The fully equipped 24 DAC-Channels version and the 12 DAC-Channels version which has only one DAC-Board (Lower DAC-Board) with the DAC-Channels 1 to 12. This “Programmer’s Manual” documents the fully equipped 24 DAC-Channels version. For the 12 DAC-Channels version some options and numbers are different than indicated in this manual, since the Higher DAC-Board with the DAC-Channels 13 to 24 are missing.

The LNHR DAC Ila can be controlled remotely from a host computer by sending simple ASCII commands, either via the serial-port (RS-232) or via the Telnet-port (Ethernet). Such ASCII commands can be created and send by all common program-languages as LabVIEW, C, Python, MATLAB, Igor, etc. After the LNHR DAC Ila has received and processed a command, it always answers with an ASCII character (error code). It is mandatory that user written control software awaits and interprets the response before a next command is send to the device – this is called handshaking.

For testing and evaluating these ASCII commands they can be sent by a standard terminal program such as PuTTY (www.putty.org). A command strings must be terminated by a <CR+LF> which is <Ctrl-J> (followed by the ENTER key) when using PuTTY.

For a simple test you can send the command “IDN?” to the device and then you must receive the hardware information of the connected LNHR DAC Ila. All commands are case insensitive (“IDN?” = “idn?”).

3 Communication with the LNHR DAC IIa

The communication between the host computer (PC) and the LNHR DAC IIa can be either via the serial communication (RS-232) or via the local area network (LAN) communication (TCP/IP Telnet) using the Ethernet standard with a speed of 100 Mbps or 1 Gbps (Gigabit Ethernet).

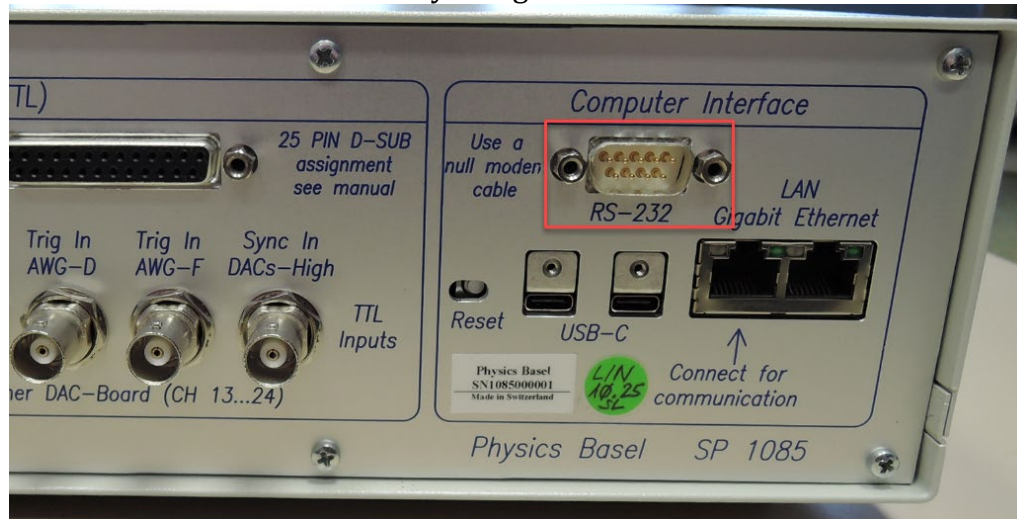
Note: For both communication channels handshaking between the LNHR DAC IIa and the control software is required to ensure reliable data transmission. Proper handshaking is implemented in the LabVIEW drivers which are included on the USB-stick.

It is also allowed to have both communication channels open and send command either via RS-232 or TCP/IP Telnet. Furthermore, the *LNHR DAC IIa Commander* (PC GUI application to control the DAC IIa) can run in parallel without any problems.

3.1 Communication via Serial-Port (RS-232)

Communication by RS-232 (<https://en.wikipedia.org/wiki/RS-232>) is a simple and robust point to point serial bidirectional data transmission.

The serial port of host computer (PC) must be connected to the 9-pin D-Sub connector on the back of the LNHR DAC IIa by using a null modem female-female D-Sub 9-pin cable:



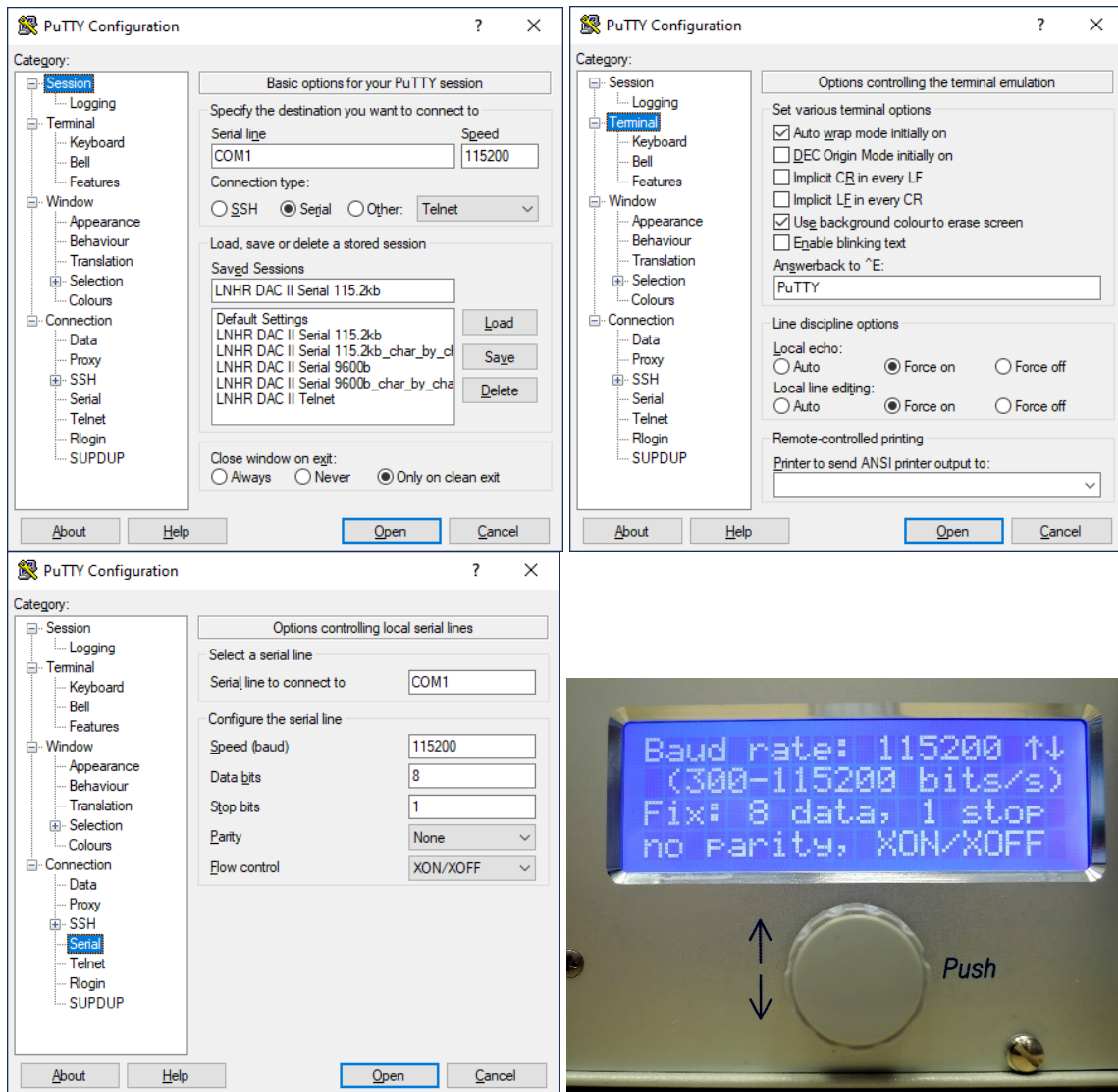
Different Baud rates from 300 to 115'200 bit/sec can be selected locally on the device under the menu item "RS-232 Settings". The following Baud rates can be selected: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bit/sec. At delivery the Baud rate is set to 9'600 bit/sec. Make sure that the speed of the serial port (Baud rate) on the host computer (PC) is the same as configured on the LNHR DAC IIa.

The number of data bits is fixed to eight with one stop bit and the parity is not used. The data flow is controlled via the XON/XOFF protocol.

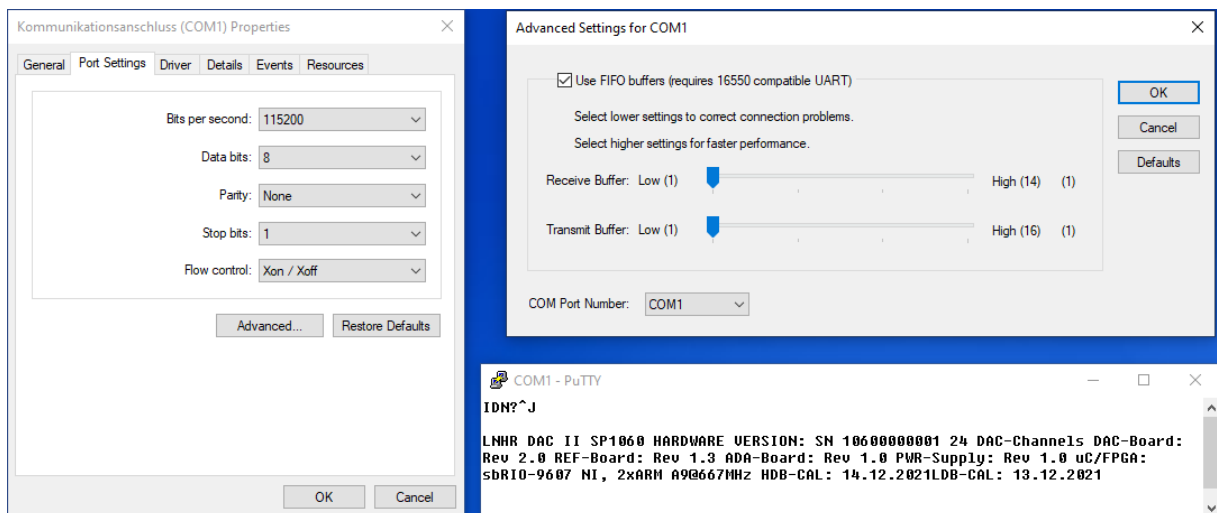
Note: A RS-232 command strings must be terminated by a Line-Feed <LF> which is <Ctrl-J> (followed by the ENTER key) when using PuTTY in its standard configuration.

When using the highest communication speed of 115.2 kbit/s the cable length shouldn't exceed 3 meters – otherwise, erroneous communication may occur.

Here are the typical COM1 port settings in PuTTY and on the LNHR DAC IIa while using the highest communication speed of 115.2 kbit/s:



At a communication speed of 115.2 kbit/s the properties of the communication port (COM1) and its advanced settings should be like this on the host computer (Windows 10 PC):



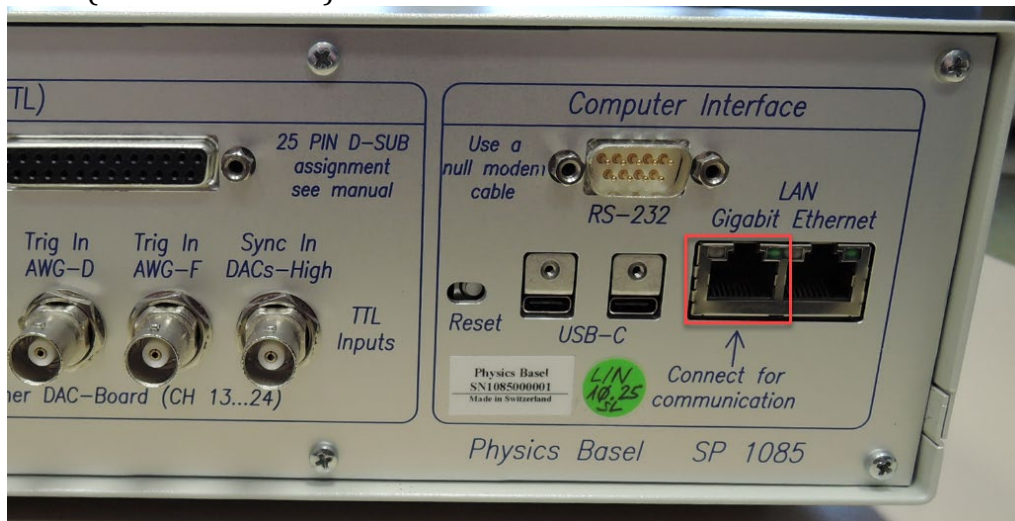
3.2 Communication via TCP/IP-Telnet Port

The communication via Telnet (en.wikipedia.org/wiki/Telnet) is an old, simple and robust bidirectional data transmission via TCP/IP (Ethernet LAN). Telnet makes a connection to Transmission Control Protocol (TCP) port number 23.

The lack of Telnet security is reduced by operating the LNHR DAC IIa in a private network which is strongly recommended (https://en.wikipedia.org/wiki/Private_network). Normally it is configured in a private network in the TCP/IPv4-address range from 192.168.0.0 to 192.168.255.255 (256 contiguous class C networks). At delivery the IP-Address is set to 192.168.0.5 and the Subnet-Mask to 255.255.255.0, but it can be modified locally on the device under the menu item “TCP/IP Settings”.

Note: A Telnet command strings must be terminated by a Carriage-Return <CR> and a Line-Feed <LF> which PuTTY automatically sends when the ENTER key is pressed.

The private network interface card of host computer (PC) must be connected to the left Gigabit Ethernet connector on the back of the LNHR DAC IIa by using a high-quality LAN cable (minimum Cat 5e):

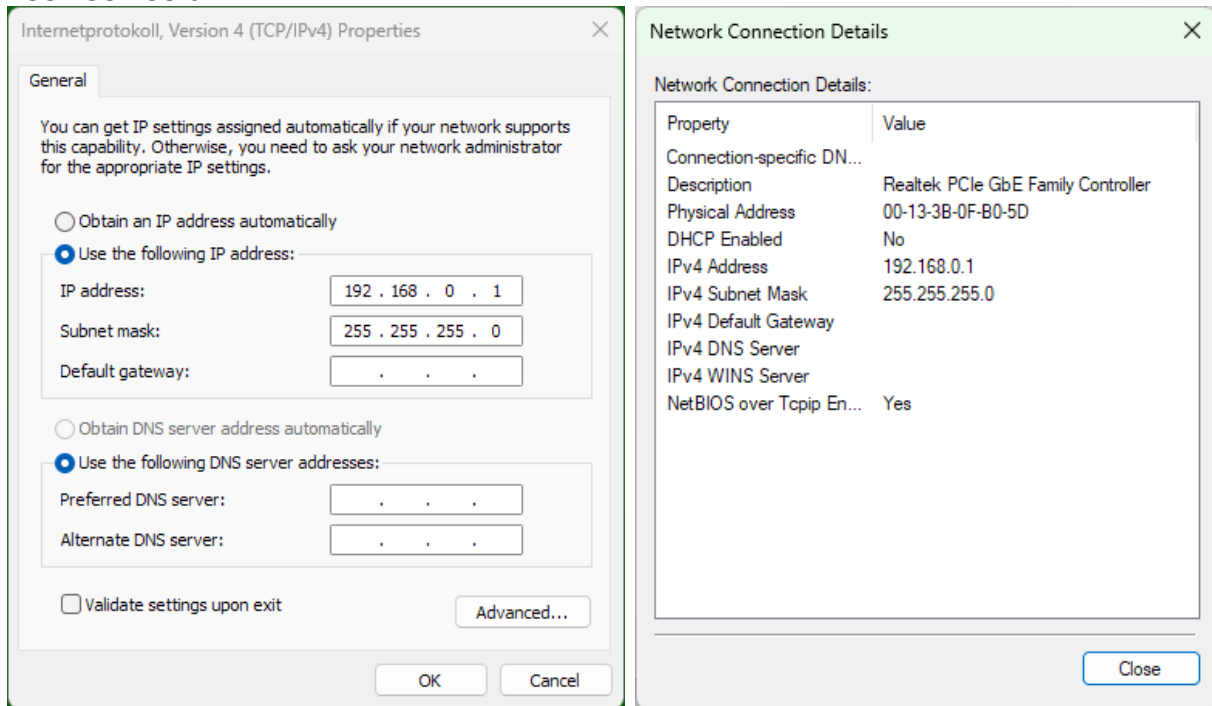


The data rate of 10 Mbps, 100 Mbps or 1 Gbps is automatically selected by the network interface card (NIC) of the host computer.

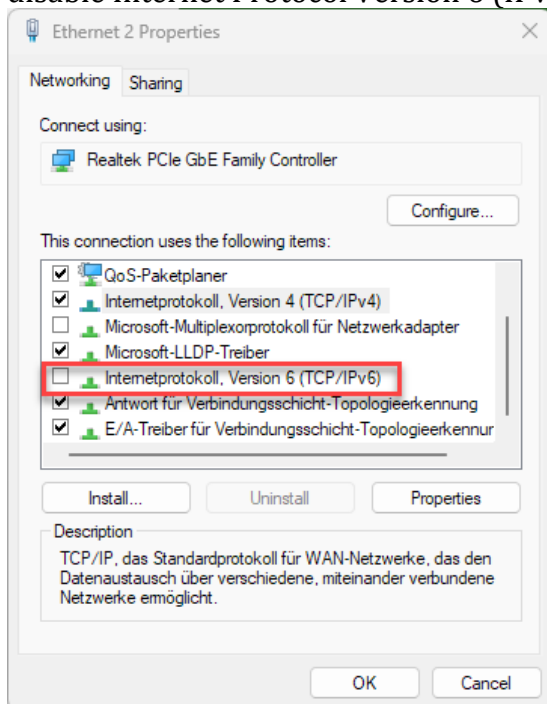
The actual data rate is indicated by the left LED on the LAN port of LNHR DAC IIa: 10 Mbps=OFF, 100 Mbps=GREEN, 1 Gbps=YELLOW.

The right LED on the LAN port shows the activity on the LAN; it is flashing when active.

Here is the typical private network configuration of the NIC on the host computer (Windows 10 PC); the host IP-Address is set to 192.168.0.1 and its Subnet-Mask also to 255.255.255.0:



Note: To avoid potential communication issues, it is strongly recommended that you disable Internet Protocol Version 6 (IPv6) on this network interface card:



Here are the typical Telnet settings in PuTTY and the standard “IP-Settings” on the LNHR DAC Ila:

The image displays three screenshots of the PuTTY Configuration dialog box and a photograph of the LNHR DAC Ila terminal display.

- Top Left Screenshot:** Shows the 'Basic options for your PuTTY session' category. The 'Host Name (or IP address)' is set to 192.168.0.5 and the 'Port' is 23. The 'Connection type' is set to 'Other: Telnet'. A 'Saved Sessions' list includes 'LNHR DAC II(a) Telnet'.
- Top Right Screenshot:** Shows the 'Options controlling the terminal emulation' category. Under 'Set various terminal options', 'Auto wrap mode initially on' and 'Use background colour to erase screen' are checked. Under 'Line discipline options', 'Local echo' and 'Local line editing' are both set to 'Auto'.
- Bottom Left Screenshot:** Shows the 'Options controlling Telnet connections' category. Under 'Telnet protocol adjustments', 'BSD (commonplace)' is selected for 'Handling of OLD_ENVIRON ambiguity'. Under 'Telnet negotiation mode', 'Passive' is selected. A red arrow points to the 'Passive' radio button. A text box below reads: **Important: Prevents from sending negotiating sequence at startup!**
- Bottom Right Screenshot:** A photograph of the LNHR DAC Ila terminal display showing the following text:


```
TCP/IP-Address:
  192.168.000.005
Subnet-Mask:
  255.255.255.000
```

 Below the display is a circular button labeled 'Push' with a vertical double-headed arrow next to it.

4 Grouping of the Commands

The grouping of the command is based on the *Tabs* used by the *LNHR DAC IIa Commander* which is the PC application to control the LNHR DAC IIa. For further information see the “LNHR DAC IIa Commander Description”. When using the *LNHR DAC IIa Commander* the last line of the “Tip Strip” (which pops up when the mouse is on a control or indicator) shows the corresponding remote command string. SET Commands are indicated in square brackets [SET] while QUERY Commands are shown in curly brackets {QUERY}.

The detailed explanations of all the LNHR DAC IIa commands are structured into the following four chapters from 5 to 8 – the *Tab Numbers* referenced to the *LNHR DAC II Commander* are also indicated:

5. SET Commands (Fast Write) – *Tabs 1, 2 and 4*

- 5.1 SET DAC Commands – *Tab 1*
- 5.2 SET AWG Commands – *Tab 4*
- 5.3 SET WAVE Commands – *Tab 4*
- 5.4 SET POLYNOMIAL Commands – *Tab 2*

Write to the DAC (Value, Status, Bandwidth), to the AWG-Memory, to the WAV-Memory and to the Polynomials-Coefficients. SET Commands can be repeated at a maximum rate of around 1 kHz (1 msec).

6. Multiple SET Command (Fast Write) – *Tabs 1, 2 and 4*

Several SET Commands can be combined to a Multiple SET Command string which is more efficient than sending several single SET Commands. With such a Multiple SET Command updating all the 24 DAC-Channels can be done withing only around 3.6 msec.

7. QUERY Commands (Fast Read) – *Tabs 1, 2, 3, 4 and 7*

- 7.1 QUERY DATA Commands – *Tabs 1, 2, 3 and 4*
- 7.2 QUERY INFORMATION Commands – *Tabs 1, 7*

Read from the DAC (Value, Status, Bandwidth, Mode, AWG-Memory, WAV-Memory, Polynomials-Coefficients, Information).

QUERY Commands can be repeated at a maximum rate of around 1 kHz (1 msec).

8. CONTROL Commands (Read / Write) – *Tabs 1, 2, 3 and 4*

- 8.1 DAC Update-Mode & Synchronization CONTROL Commands – *Tab 1*
- 8.2 RAMP/STEP-Generator CONTROL Commands – *Tab 2*
- 8.3 2D-Scan CONTROL Commands – *Tab 2*
- 8.4 AWG CONTROL Commands – *Tab 3*
- 8.5 Standard Waveform Generation (SWG) CONTROL Commands – *Tab 4*
- 8.6 Wave CONTROL Commands – *Tab 4*

Read / Write access to numerous controls of the device (DAC Update Mode, RAMP/STEP-Generators, AWGs, Standard Waveform Generation).

CONTROL Commands can be repeated at a maximum rate of around 100 Hz (10 msec).

5 SET Commands (Write)

The LNHR DAC Ila can be fast controlled remotely from a host computer by sending SET Commands, which are ASCII strings terminated by a <LF>. The SET Commands allow a fast write-access to the device.

SET Commands can be repeated at a maximum rate of around 1 kHz (1 msec). For such high update rates, it is mandatory that the response (error-code) of the device is read and interpreted before a next SET Command is transmitted. This handshaking allows a stable and reliable data transfer from the host computer to the device.

For even faster remote control, several single SET Commands can be combined to a multiple SET Command string which allows updating all the 24 DAC-Voltages within only 3.6 msec (for details see chapter “Multiple SET Commands”).

The DAC-Value (Output Voltage), the DAC-Status (ON/OFF) and the DAC-Bandwidth (LBW=100 Hz, HBW=100 kHz) can be set with these commands. Further the six AWG-Memories (AWG-A/B/E/C/D/F), the seven WAV-Memories (WAV-A/B/E/C/D/F/S) and the four Polynomials (POLY-A/B/C/D) can be programmed. The WAV-S holds a waveform saved by the user.

The detailed description of all the SET Commands is grouped in the following four sections:

5.1 SET DAC Commands: Set the DAC-Value, the DAC-Status and the DAC-Bandwidth

5.2 SET AWG Commands: Program the AWG-Memories (AWG-A/B/E/C/D/F)

5.3 SET WAV Commands: Program the Wave-Memories (WAV-A/B/E/C/D/F/S)

5.4 SET POLY Commands: Set the Polynomials-Coefficients (POLY-A/B/C/D)

5.1 SET DAC Commands

With these SET DAC Commands, the registered DAC-Values (0x000000...0xFFFFFFFF), the DAC-Status (ON/OFF) and the DAC-Bandwidth (LBW, HBW) can be set.

The registered DAC-Value is immediately written to the selected DAC-Channel and the DAC-Voltage gets updated, if the MODE is “DAC”. If the MODE is “SYN” a synchronous update of all DAC-Channels is performed by an internal (command) or external (TTL input) SYNC-event. For further details see the chapter “DAC Update-Mode & Synchronization CONTROL Commands” in the section of the CONTROL Commands.

5.1.1 SET a DAC-Channel to a registered DAC-Value (HEX)

```
"DAC-CH[1...24] <SPACE> DAC-Value[0x000000...0xFFFFFFFF] <LF>"
```

Examples:

```
SET DAC-Channel 1 to 0 V: "1 7FFFFFFF<LF>"
```

```
SET DAC-Channel 2 to +1 V: "2 8CCCCC<LF>"
```

```
SET DAC-Channel 3 to -2.5 V: "3 600000<LF>"
```

```
SET DAC-Channel 18 to +3.4 V: "18 AB851E<LF>"
```

These DAC-Values (HEX) corresponds to the following DAC-Voltages (± 10 V) – see also the chapter “Converting DAC-Voltage to DAC-Value”:

```
0x7FFFFFFF = 0 V
```

```
0x8CCCCC = +1 V
```

0xBFFFFFF = +5 V
 0xFFFFFFFF = +10 V
 0x733333 = -1 V
 0x400000 = -5 V
 0x000000 = -10 V

5.1.2 SET ALL DAC-Channels to a registered DAC-Value (HEX)

"ALL <SPACE> DAC-Value[0x000000...0xFFFFFFFF] <LF>"

Examples:

SET ALL DAC-Channels to 0 V: "ALL 7FFFFFF<LF>"

SET ALL DAC-Channels to -5 V: "ALL 400000<LF>"

5.1.3 SET a DAC-Channel Status (ON or OFF)

"DAC-Channel[1...24] <SPACE> Status[ON/OFF] <LF>"

Each DAC-Channel can be individually be switched ON or OFF; after power-up all DAC-Channels are switched OFF. In the OFF-Status the LED on the front-panel is turned off and the output is passively pulled-down to AGND via 1 M Ω resistor in parallel with a 22 nF capacitor. In the ON-Status the LED on the front-panel is turned on and the color indicates the selected Bandwidth:

Yellow = LBW (100 Hz), Blue = HBW (100 kHz)

Examples:

SET DAC-Channel 1 ON: "1 ON<LF>"

SET DAC-Channel 15 OFF: "15 OFF<LF>"

5.1.4 SET ALL DAC-Channels to a Status (ON or OFF)

"ALL <SPACE> Status[ON/OFF] <LF>"

Examples:

SET ALL DAC-Channels ON: "ALL ON<LF>"

SET ALL DAC-Channels OFF: "ALL OFF<LF>"

5.1.5 SET a DAC-Channel Bandwidth (LBW=100 Hz, HBW=100 kHz)

"DAC-Channel[1...24] <SPACE> Bandwidth[LBW/HBW] <LF>"

For each DAC-Channel two different Bandwidths can be selected: Low Bandwidth (LBW) corresponds to 100 Hz (-3 dB), High Bandwidth (HBW) corresponds to 100 kHz (-3 dB) The ON-Status LED on the front-panel indicates the selected Bandwidth:

Yellow = LBW, Blue = HBW

CAUTION: To prevent from glitch-voltages, first switch OFF the corresponding DAC-Channel before switching the Bandwidth!

Examples:

SET DAC-Channel 6 LBW=100 Hz: "6 LBW<LF>"

SET DAC-Channel 17 HBW=100 kHz: "17 HBW<LF>"

5.1.6 SET ALL DAC-Channels Bandwidth (LBW=100 Hz, HBW=100 kHz)

"ALL <SPACE> Bandwidth[LBW/HBW] <LF>"

CAUTION: To prevent from glitch-voltages, first switch OFF ALL the DAC-Channels before switching ALL the Bandwidths!

Examples:

SET ALL DAC-Channels to LOW Bandwidth (LBW=100 Hz): "ALL LBW<LF>"

SET ALL DAC-Channels to HIGH Bandwidth (HBW=100 kHz): "ALL HBW<LF>"

After a SET DAC Command has been received and processed successfully, the device responds with zero ("0") followed by a <CR+LF>. This "No error" response must be awaited before a next SET DAC Command can be send. If a SET DAC Command cannot be interpreted an Error-Code followed by a <CR+LF> is returned. The meaning of the SET DAC Command "Error-Codes" is the following:

"0" = No error (normal)

"1" = Invalid DAC-Channel

"2" = Missing DAC-Value, Status or BW

"3" = DAC-Value out of range

"4" = Mistyped

"5" = Writing not allowed (Ramp/Step-Generator or AWG are running on this DAC-Channel)

5.2 SET AWG Commands

The six AWG-Memories (AWG-A/B/E/C/D/F) can be directly programmed by using the SET AWG Commands. The size of each AWG-Memory is 65'000 points corresponding to an address-range (HEX) from 0x0000 to maximum 0xFDE7 (=64'999). Each address holds a 24-bit DAC-Value in the range from 0x000000 (-10 V) to 0xFFFFFFFF (+10 V). After power-up all AWG-Memories are completely filled with the DAC-Value 0x7FFFFFFF, corresponding to zero (0 V) DAC-Voltage.

Since these AWG-Memories are located on the FPGA no linearization can be applied on the SET AWG Commands. If linearization is needed, the corresponding Wave-Memory has to be filled first. While transferring the Wave-Memory to the AWG-Memory the linearization for the dedicated DAC-Channel can be applied.

The AWG-Memories A, B and E are associated to the Lower DAC-Board (CH 1...12) and the AWG-Memories C, D and F to the Higher DAC-Board (CH 13...24). This AWG naming was chosen to ensure backwards compatibility with the old LNHR DAC II drivers.

CAUTION: It is not allowed to set AWG-Values while the same AWG is running – the AWG has to be stopped before programming new content!

5.2.1 SET an AWG-Memory Address (HEX) to an AWG-Value (HEX)

```
"AWG-Memory[AWG-A/B/E/C/D/F] <SPACE> AWG-Address[0x0000...0xFDE7]
<SPACE> AWG-Value[0x000000...0xFFFFFFFF] <LF>"
```

These AWG-Values (HEX) corresponds to the following DAC-Voltages (± 10 V) – see also chapter "Converting DAC-Voltage to DAC-Value":

0x7FFFFFFF = 0 V

0x8CCCCC = +1 V

0xBFFFFFFF = +5 V

0xFFFFFFFF = +10 V

0x 733333 = -1V

0x400000 = -5 V
 0x000000 = -10 V

Examples:

SET AWG-Memory A at Address 0x0000 to 0 V: "AWG-A 0000 7FFFFFFF<LF>"
SET AWG-Memory B at Address 0x0025 to +1 V: "AWG-B 0025 8CCCCC<LF>"
SET AWG-Memory F at Address 0x84CF to +10 V: "AWG-F 84CF FFFFFFFF<LF>"

5.2.2 SET ALL AWG-Memory Addresses to an AWG-Value (HEX)

"AWG-Memory[AWG-A/B/E/C/D/F] <SPACE> ALL <SPACE>
 AWG-Value[0x000000...0xFFFFFFFF] <LF>"

The complete AWG-Memory [AWG-A/B/E/C/D/F], starting from the AWG-Address of 0x0000 till the maximum AWG-Address of 0xFDE7 (=64'999), is filled with the same AWG-Value [0x000000...0xFFFFFFFF]. This takes around two seconds.

Examples:

SET ALL AWG-Values of AWG-Memory A to 0 V: "AWG-A ALL 7FFFFFFF<LF>"
SET ALL AWG-Values of AWG-Memory F to +5 V: "AWG-F ALL BFFFFFFF<LF>"

After a SET AWG Command has been received and processed successfully (takes around two seconds), the device responds with zero ("0") followed by a <CR+LF>. This "No error" response must be awaited before a next SET AWG Command can be send. If the SET AWG Command cannot be interpreted, an Error-Code with a <CR+LF> is returned; the meaning of the SET AWG Command "Error-Codes" is the following:

"0" = No error (normal)
 "1" = Invalid AWG-Memory
 "2" = Missing AWG-Address and/or AWG-Value
 "3" = AWG-Address and/or AWG-Value out of range
 "4" = Mistyped

5.3 SET WAVE Commands

The seven Wave-Memories (WAV-A/B/E/C/D/F/S) can be programmed by using the SET WAV Commands. The size of each Wave-Memory is maximum 65'000 points corresponding to an address-range (HEX) from 0x0000 to maximum 0x84CF (=33'999). Each address holds a DAC-Voltage in the range from -10 V to +10 V. After power-up all WAV-Memories are cleared (no entries).

The Wave-Memory can be fast copied to the corresponding AWG-Memory. During this copy-process the user programmable polynomial (POLY-A/B/C/D) can be applied; further the linearization for the selected DAC-Channel can be turned on. These options allow fast and precise adaptive 2D-Scans on up to four channels.

The Wave-Memories A, B and E are dedicated to the AWG-Memories A, B and E (Lower DAC-Board) and the Wave-Memories C, D and F to the AWG-Memories C, D and F (Higher DAC-Board). This AWG naming was chosen to ensure backwards compatibility with the old LNHR DAC II drivers. The Wave-Memory SAVED (WAV-S) holds a waveform saved by the user.

Note: Writing large Wave-Memories (>5'000 points) remotely can be very time and resource consuming! It is strongly recommended to use the Standard Wave Generator

(SWG) to program large Wave-Memories – see Chapter “Standard Waveform Generation (SWG) CONTROL Commands”.

5.3.1 SET a WAV-Memory Address (HEX) to a WAV-Voltage (Volt)

```
"WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> WAV-Address[0x0000..0xFDE7]
<SPACE> WAV-Voltage[±10.000000] <LF>"
```

The WAV-Voltage is a floating-point number in the range between -10.000000 V and +10.000000; the decimal point must be a period (point).

Note: Writing a single WAV-Voltage to an empty (cleared) WAV-Memory will fill the WAV-Memory starting from WAV-Address 0x0000, up to the selected WAV-Address, with the same WAV-Voltage. If a next WAV-Voltage is written at a higher WAV-Address where the WAV-Memory it is still empty, it will fill the gap with this next WAV-Voltage. This feature enables step functions to be generated easily at different levels.

To fill an empty (cleared) WAV-Memory completely with a certain WAV-Voltage, simply write that voltage to the highest WAV-Address, 0xFDE7 (=64'999).

Examples:

```
SET WAV-Memory A at Address 0x0000 to 0 V:      "WAV-A 0000 0<LF>"
SET WAV-Memory E at Address 0x12AA to +1.234567 V: "WAV-E 12AA 1.234567<LF>"
SET WAV-Memory C at Address 0x84CF to -8.881717 V: "WAV-C 84CF -8.881717 <LF>"
```

5.3.2 SET ALL WAV-Memory Addresses to a WAV-Voltage (Volt)

```
"WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> ALL <SPACE>
WAV-Voltage[±10.000000] <LF>"
```

The complete Wave-Memory [WAV-A/B/E/C/D/F/S], starting from the WAV-Address of 0x0000 till the maximum WAV-Address of 0xFDE7 (=64'999), is filled with the same WAV-Voltage in the range between -10.000000 V and +10.000000. The decimal point must be a period (point). Any existing voltages will be overwritten. This takes around one seconds.

Examples:

```
SET ALL WAV-Voltage of WAV-Memory S (saved) to 0 V:  "WAV-S ALL 0<LF>"
SET ALL WAV-Voltages of WAV-Memory B to +5.123456 V: "WAV-B ALL 5.123456<LF>"
```

After a SET WAV Command has been received and processed successfully (takes around one seconds), the device responds with zero ("0") followed by a <CR+LF>. This “No error” response must be awaited before a next SET WAV Command can be send. If the SET WAV Command cannot be interpreted, an Error-Code with a <CR+LF> is returned; the meaning of the SET WAV Command “Error-Codes” is the following:

```
"0" = No error (normal)
"1" = Invalid WAV-Memory
"2" = Missing WAV-Address and/or WAV-Voltage
"3" = WAV-Address and/or WAV-Voltage out of range
"4" = Mistyped
```

5.4 SET POLYNOMIAL Commands

The coefficients of four polynomials (POLY-A/B/C/D) can be programmed by using the SET POLY Commands. The coefficients are in ascending order; starting with a_0 (constant) followed by a_1 (*x), a_2 (*x²), a_3 (*x⁴) and so on. The order of the polynomial is not restricted, but is normally smaller than ten.

The polynomial is applied when the Wave-Memory (Voltage) is written to the AWG-Memory, but only when the corresponding Control “Apply Polynomial (A/B/C/D)” is set. Then the polynomial is calculated on the WAV-Voltage function and therefore an a_0 -coefficient of 1 adds 1 Volt to the AWG-Function.

Since the polynomial coefficients can be fast updated by the host-PC, a simple and efficient fully adaptive 2D-Scan can be implemented.

This is done by using the linear STEP-Generator (“Step @AWG Stop (A/B/C/D)”) in combination with a predefined AWG-Function (Ramp), while the Controls “Reload AWG MEM (A/B/C/D)” and “Apply Polynomial (A/B/C/D)” must be set.

After each step of the linear STEP-Generator the AWG-Memory gets reloaded from the WAV-Memory by applying the actual polynomial, which can be adapted by the user.

If “Auto-Start AWG” is set, the modified AWG-Function gets started automatically after the AWG-Memory was updated. Since the stop of the AWG-Function triggers the next step of the linear STEP-Generator, the scan automatically runs through all the linear steps until finished. A minimum time delay of 5 msec is implemented from the update of the STEP-Generator to the restart of the AWG.

A linear adaptive 2D-Scan can be done by using the “Adaptive Shift Voltage (per Step)”. With this parameter a linear adaptation of the y-axis based on the x-axis can simply be implemented. After each step of the STEP-Generator this “Adaptive Shift Voltage” gets applied to the AWG function by linearly modifying the polynomial coefficient a_0 (constant) based on the Cycles-Done in x-axis.

The “Adaptive Shift Voltage (per Step)” can be read/write by the user – see the chapter “2D-Scan CONTROL Commands”.

5.4.1 SET the POLYNOMIAL Coefficients

```
"Polynomial[POLY-A/B/C/D] <SPACE> a0 <SPACE> a1 <SPACE> ... an<LF>"
```

The polynomial coefficients $a_0, a_1...a_n$ are in ascending order and are floating-point/exponential (E) numbers. The decimal point must be a period (point).

Examples:

SET Polynomial A to $a_0=0, a_1=1.25, a_2=1E-3, a_3=2.3E-6$: "POLY-A 0 1.25 1E-3 2.3E-6<LF>"

SET Polynomial C to $a_0=0.235, a_1=1, a_2=3E-4, a_3=8.1E-9$: "POLY-C 0.235 1 3E-4 8.1E-9<LF>"

After a SET POLY Command has been received and processed successfully, the device responds with zero ("0") followed by a <CR+LF>. This “No error” response must be awaited before a next SET POLY Command can be send. If the SET POLY Command cannot be interpreted, an Error-Code with a <CR+LF> is returned. The meaning of the SET POLY Command “Error-Codes” is the following:

"0" = No error (normal)

"1" = Invalid Polynomial Name

"2" = Missing Polynomial Coefficient(s)

"4" = Mistyped

6 Multiple SET Command (Write)

Up to thousand (1'000) different single SET Commands can be combined in a multiple SET command string. Each SET command must be separated by a semicolon (";") and the end of the string must be terminated with a <LF>. For each of the single SET command, its own error-code is generated, separated by a semicolon (";") and then terminated by a <CR+LF>. For example, a multiple SET command string holding 24 SET commands (e.g., to set all the 24 DAC-Channels) also 24 error-codes are generated. Make sure that all the received error-codes are "0" (no error).

A multiple DAC Command string allows DAC voltages to be updated with a smaller time-skew than using single SET Commands which can be repeated at a maximum rate of around 1 kHz (1 msec). Updating all 24 DAC-Channels by using 24 single SET Commands would last at least 24 msec.

Using a multiple SET command string, DAC update rates of typical 6.6 kHz (150 μ s) are reached. All the 24 DAC-Channels can be updated within only around 3.6 msec when sending a multiple SET command string. The time-skew from updating the first DAC-Channel (1) to the last DAC-Channel (24) is around 3.6 msec.

If a smaller time-skew is needed, the synchronous DAC Mode (SYNC) must be selected; see the chapter "DAC Update-Mode & Synchronization CONTROL Commands".

Note: Different SET Commands can be mixed in a multiple SET command string, but the SET WAVE and the SET POLYNOMIAL Commands are excluded from mixing with other Commands. Nevertheless, multiple SET WAVE or multiple SET POLYNOMIAL Commands are allowed.

Examples:

SET DAC-CH 1 to +1V; DAC-CH 2 to +2V; DAC-CH 3 to +3V; DAC-CH 4 to +4V; DAC-CH 5 to +5V; DAC-CH 6 to +6V; DAC-CH 7 to +7V; DAC-CH 8 to +8V; DAC-CH 9 to +9V; DAC-CH 10 to +10V; DAC-CH 11 to -1V; DAC-CH 12 to -2V:

```
"1 8CCCCC;2 999999;3 A66666;4 B33332;5 BFFFFFF;6 CCCCCC;7 D99999;8 E66665;
9 F33332;10 FFFFFFF;11 733333;12 666666<LF>"
→ "0;0;0;0;0;0;0;0;0;0;0;0<CR+LF>"
```

SET DAC-CH 3 ON; DAC-CH 3 to +1V; DAC-CH 14 to +5V; DAC-CH 4 to -5V; DAC-CH 4 HBW; DAC-CH 4 ON:

```
"3 ON;3 8CCCCC;14 BFFFFFF;4 400000;4 HBW;4 ON<LF>"
→ "0;0;0;0;0;0<CR+LF>"
```

SET Wave-Memory A at Address 0x0000 to -1.2 V; SET Wave-Memory A at Address 0x0001 to -0.9 V; SET Wave-Memory A at Address 0x0002 to -0.6 V:

```
"WAV-A 0000 -1.2;WAV-A 0001 -0.9;WAV-A 0002 -0.6<LF>"
→ "0;0;0<CR+LF>"
```

Note: The RS-232 receive-buffer on the LNHR DAC IIa has fixed size of 128 bytes which is given by the real-time Linux running on the device. To avoid a possible buffer-overflow while using multiple SET Commands via the serial communication (RS-232) it is recommended to restrict the size of the command to a maximum of 125 characters.

Sending longer multiple SET Commands via RS-232 can be truncated and result in an error-code response from the LNHR DAC IIa. The probability of such a RS-232 buffer-overflow depends on the workload of LNHR DAC IIa (see CPU-Load). The heavier the workload, the higher the chance for a buffer-overflow on the serial port.

There is no such limitation when using the Ethernet bus (TCP/IP-Telnet) to send long multiple SET Commands.

7 QUERY Commands (Read)

The LNHR DAC IIa can be readout by the host computer by sending QUERY Commands. A QUERY Command always ends with a question-mark ("?") followed by the termination character <LF>. The QUERY Commands allow a fast read-access to the device.

Only single QUERY Commands can be released. If a QUERY Command cannot be interpreted, the device response with a question-mark ("?") followed by a <CR+LF>. If the readout consists of several DATA-values, the values are separated by a semicolon ";".

QUERY Commands can be repeated at a maximum rate of around 1 kHz (1 msec). But it is mandatory that the response (DATA/INFORMATION or "?") of the device is read, before a next QUERY Command is transmitted.

The detailed description of all QUERY Commands is grouped in the following two sections:

7.1 QUERY DATA Commands: Readout the Actual DAC-Value, the Registered DAC Value, the DAC-Status, the DAC-Bandwidth, the DAC-MODE, the AWG-Memories, the WAV-Memories and the Polynomial Coefficients.

7.2 QUERY INFORMATION Commands: Readout several information, settings and status from the device.

7.1 QUERY DATA Commands (Read)

The QUERY DATA Commands allows the readout of several values which can also be written by the SET Commands; with expectation of the DAC-MODE which is set by a CONTROL Command.

7.1.1 QUERY an Actual DAC-Value (HEX)

```
"DAC-Channel[1...24] <SPACE> V?<LF>"
→ DAC-Value[0x000000...0xFFFFFFFF]<CR+LF>
```

The Actual DAC-Value (HEX) corresponds to the value that is loaded into the DAC and is therefore also present as a DAC output voltage.

Examples:

```
QUERY Actual DAC-Value of Channel 1: "1 V?<LF>"
→ "7FFFFFF<CR+LF>"
```

```
QUERY ACTUAL DAC-Value of Channel 18: "18 V?<LF>"
→ "AB851E<CR+LF>"
```

7.1.2 QUERY ALL Actual DAC-Values (HEX)

```
"ALL <SPACE> V?<LF>"
→ DAC-Value 1;DAC-Value 2;... ;DAC-Value 24[0x000000...0xFFFFFFFF]<CR+LF>
```

Example:

```
QUERY ALL Actual DAC-Values: "ALL V?<LF>"
→ "7FFFFFF;7FFFFFF;BFFFFFF;400000;FFFFFF;000000;7FFFFFF;AAAAAA;
7FFFFFF;7FFFFFF;7FFFFFF;7FFFFFF;FFFFFF;000000;7FFFFFF;BBBBB;
7FFFFFF;AB851E;7FFFFFF;7FFFFFF;FFFFFF;000000;7FFFF;CCCCCC;<CR+LF>"
```

7.1.3 QUERY a Registered DAC-Value (HEX)

```
"DAC-Channel[1...24] <SPACE> VR?<LF>"
```

→ DAC-Value[0x000000...0xFFFFFFFF]<CR+LF>

The Registered DAC-Value (HEX) corresponds to the value that is registered to be loaded to the DAC when a SYNC-event arrives; this may not be the actual DAC output voltage.

Examples:

QUERY Registered DAC-Value of Channel 1: "1 VR?<LF>"

→ "7FFFFFFF<CR+LF>"

QUERY Registered DAC-Value of Channel 18: "18 VR?<LF>"

→ "AB851E<CR+LF>"

7.1.4 QUERY ALL Registered DAC-Values (HEX)

"ALL <SPACE> VR?<LF>"

→ DAC-Value 1;DAC-Value 2;... ;DAC-Value 24[0x000000...0xFFFFFFFF]<CR+LF>

Example:

QUERY ALL Registered DAC-Values: "ALL VR?<LF>"

*→ "7FFFFFFF;7FFFFFFF;BFFFFFFF;400000;FFFFFFF;000000;7FFFFFFF;AAAAAA;
7FFFFFFF;7FFFFFFF;7FFFFFFF;7FFFFFFF;FFFFFFF;000000;7FFFFFFF;BBBBB;
7FFFFFFF;AB851E;7FFFFFFF;7FFFFFFF;FFFFFFF;000000;7FFFFF;CCCCCC;<CR+LF>"*

7.1.5 QUERY a DAC-Status (ON or OFF)

"DAC-Channel[1...24] <SPACE> S?<LF>"

→ Status[ON/OFF]<CR+LF>

Examples:

QUERY DAC-Status of Channel 1: "1 S?<LF>"

→ "ON<CR+LF>"

QUERY DAC-Status of Channel 15: "15 S?<LF>"

→ "OFF<CR+LF>"

7.1.6 QUERY ALL DAC-Status (ON or OFF)

"ALL <SPACE> S?<LF>"

→ Status 1;Status 2;... ;Status 24[ON/OFF]]<CR+LF>

Example:

QUERY ALL DAC-Status: "ALL S?<LF>"

*→ "ON;OFF;ON;OFF;ON;ON;ON;OFF;ON;OFF;ON;OFF;ON;ON;ON;OFF;ON;OFF;ON;
OFF;ON;ON;ON;OFF<CR+LF>"*

7.1.7 QUERY a DAC-Channel Bandwidth (LBW=100 Hz or HBW=100 kHz)

"DAC-Channel[1...24] <SPACE> BW?<LF>"

→ BW[LBW/HBW]<CR+LF>

Examples:

QUERY the Bandwidth of DAC-Channel 6: "6 BW?<LF>"

→ "LBW<CR+LF>"

QUERY the Bandwidth of DAC-Channel 17: "17 BW?<LF>"

→ "HBW<CR+LF>"

7.1.8 QUERY ALL DAC-Channel Bandwidths (LBW=100 Hz or HBW=100 kHz)

```
"ALL <SPACE> BW?<LF>"
→ BW 1;BW 2;...;BW 24[LBW/HBW]<CR+LF>
```

Example:

```
QUERY ALL DAC-Channel Bandwidths: "ALL BW?<LF>"
→ "HBW;LBW;LBW;LBW;LBW;LBW;LBW;LBW;HBW;LBW;LBW;LBW;LBW;LBW;
LBW;LBW;HBW;LBW;LBW;LBW;LBW;LBW;LBW;LBW<CR+LF>"
```

7.1.9 QUERY a DAC-MODE (ERR/DAC/SYN/RMP/AWG/ ---)

```
"DAC-Channel[1...24] <SPACE> M?<LF>"
→ MODE[ERR/DAC/SYN/RMP/AWG/---]<CR+LF>
```

Examples:

```
QUERY the MODE of DAC-Channel 6: "6 M?<LF>"
→ "DAC<CR+LF>"
```

```
QUERY the MODE of DAC-Channel 17: "17 M?<LF>"
→ "AWG<CR+LF>"
```

The meaning of the MODEs is the following:

“ERR” = An error is detected on this DAC-Channel; contact service.

“DAC” = Normal transparent DAC-mode (Instant-Mode). The written DAC-Values are immediately converted to an output voltage.

“SYN” = Synchronous DAC-mode; the written DAC-Values are REGISTERED and are applied after a SYNC-Command (internal or external) is received. This allows a synchronous update of all the 12 DAC-Channels of one DAC-Board (Lower or Higher) or both in parallel (Lower and Higher).

“RMP” = On this DAC-Channel a RAMP- or STEP-Generator is running; writing to this DAC-Channel is not allowed (Error-Code = “5”).

“AWG” = This DAC-Channel is occupied by a running AWG function; writing to this DAC-Channel is not allowed (Error-Code = “5”).

“---” = This DAC-Channel is not available. When a DAC-Board is reserved for AWG only (lower AWG time-jitter) the other DAC-Channels aren’t available. Writing to this DAC-Channel is not allowed (Error-Code = “5”).

7.1.10 QUERY ALL DAC-MODEs (ERR/DAC/SYN/RMP/AWG/ ---)

```
"ALL <SPACE> M?<LF>"
→ MODE_1;MODE_2;...;MODE_24[ERR/DAC/SYN/RMP/AWG/---]<CR+LF>
```

Example:

```
QUERY ALL MODEs of all DAC-Channels: "ALL M?<LF>"
→ "DAC;RMP;RMP;DAC;DAC;DAC;DAC;DAC;DAC;DAC;AWG;AWG;DAC;RMP
DAC;DAC;AWG;DAC;DAC;RMP;RMP;DAC;DAC;DAC<CR+LF>"
```

7.1.11 QUERY an AWG-Value (HEX) at AWG-Address (HEX)

```
"AWG-Memory[AWG-A/B/E/C/D/F] <SPACE> AWG-Address[0x0000...0xFDE7]?<LF>"
→ AWG-Value[0x000000...0xFFFFFFFF]<CR+LF>
```

For an explanation of the AWG-Memories, see chapter "SET AWG Commands".

CAUTION: It is not allowed to query AWG-Values while the same AWG is running – the AWG has to be stopped before reading!

Examples:

```
QUERY AWG-Memory A at Address 0x0000: "AWG-A 0000?<LF>"
→ "7FFFFFF<CR+LF>"
```

```
QUERY AWG-Memory B at Address 0x0025: "AWG-B 0025?<LF>"
→ "8CCCCC<CR+LF>"
```

```
QUERY AWG-Memory F at Address 0xFDE7: "AWG-F FDE7?<LF>"
→ "FFFFFF<CR+LF>"
```

7.1.12 QUERY a BLOCK (BLK) of 1'000 (decimal) AWG-Values (HEX)

```
"AWG-Memory[AWG-A/B/E/C/D/F] <SPACE> AWG-StartADR[0x0000...0xFA00]
<SPACE> BLK?<LF>"
→ AWG-Value@StartADR+0;AWG-Value@StartADR+1;AWG-Value@StartADR+2;
...; AWG-Value@Start_ADR+999<CR+LF>
```

Starting from the AWG-StartADR [0x0000...0xFA00] a BLOCK of 1'000 (decimal) AWG-Values [0x000000...0xFFFFFFFF] are sequentially readout. The AWG-StartADR can be at maximum 0xFA00 (=64'000), since it must be 999 (decimal) smaller than the maximal AWG-Address which is 0xFDE7 (=64'999). The query of an AWG BLOCK allows fast readout of AWG-Memories.

CAUTION: It is not allowed to query AWG-Values while the same AWG is running – the AWG has to be stopped before reading!

Example:

```
QUERY a BLOCK of 1'000 (decimal) AWG-B Values starting from AWG-Address 0x3ABC:
"AWG-B 3ABC BLK?<LF>"
→ "7FFFFFF;7FFFFFF;BFFFFFF;400000;FFFFFF;000000;7FFFFFF;AAAAAA;
7FFFFFF;7FFFFFF;7FFFFFF;7FFFFFF;FFFFFF;...;7FFFFFF<CR+LF>"
(Total 1'000 HEX-Values)
```

7.1.13 QUERY a WAV-Value (Voltage) at WAV-Address (HEX)

```
"WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> WAV-Address[0x0000...0xFDE7]?<LF>"
→ WAV-Voltage[±10.000000/NaN]<CR+LF>
```

The WAV-Voltage is a floating-point number in the range between -10.000000 V and +10.000000; the decimal point is a period (point). If the asked WAV-Address is empty (no entry) the response is "NaN" instead of a WAV-Voltage.

For an explanation of the WAV-Memories, see chapter "SET WAV Commands".

Examples:

```
QUERY WAV-Memory A at Address 0x0000: "WAV-A 0000?<LF>"
```

→ "0.000000<CR+LF>"
 QUERY WAV-Memory B at Address 0x0025: "WAV-B 0025?<LF>"
 → "1.234567<CR+LF>"
 QUERY WAV-Memory E at Address 0x80AB: "WAV-E 80AB?<LF>"
 → "-10.000000<CR+LF>"

7.1.14 QUERY a BLOCK (BLK) of 1'000 (decimal) WAV-Values (Voltage)

"WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> WAV-StartADR[0x0000...0xFA00]
 <SPACE> BLK?<LF>"
 → WAV-Voltage@StartADR+0;WAV-Voltage@StartADR+1;
 WAV-Voltage@Start-ADR+2; ...; WAV-Voltage@Start_ADR+999<CR+LF>

Starting from the WAV-StartADR [0x0000...0xFA00] a BLOCK of 1'000 (decimal) WAV-Voltages [$\pm 10.000000/\text{NaN}$] are sequentially readout. If the WAV-Address is empty (no entry) the response is "NaN" instead of a WAV-Voltage. The WAV-StartADR can be at maximum 0xFA00 (=64'000), since it must be 999 (decimal) smaller than the maximal WAV-Address which is 0xFDE7 (=64'999). The query of a WAV BLOCK allows fast readout of WAV-Memories.

Example:

QUERY a BLOCK of 1'000 (decimal) WAV-A Voltages starting from WAV-Address 0x2ABC:
 "WAV-A 2ABC BLK?<LF>"
 → "0.000000;1.000000;-1.000000;2.123456;-3.333333;...;-10.000000<CR+LF>"
 (Total 1'000 WAV-A Voltages)

7.1.15 QUERY the POLYNOMIAL Coefficients

"Polynomial[POLY-A/B/C/D]?<LF>"
 → "a0;a1;a2;a3;... a_n<CR+LF>"

Readout the polynomial coefficients $a_0, a_1, a_2 \dots a_n$ of the demanded Polynomial (POLY-A/B/C/D). The coefficients are floating-point/exponential (E) numbers with a period (point) as the decimal point.

Examples:

QUERY Polynomial C Coefficients: "POLY-C?<LF>"
 → "2.350000E-1;1.000000E+0;3.000000E-4;8.100000E-9<CR+LF>"

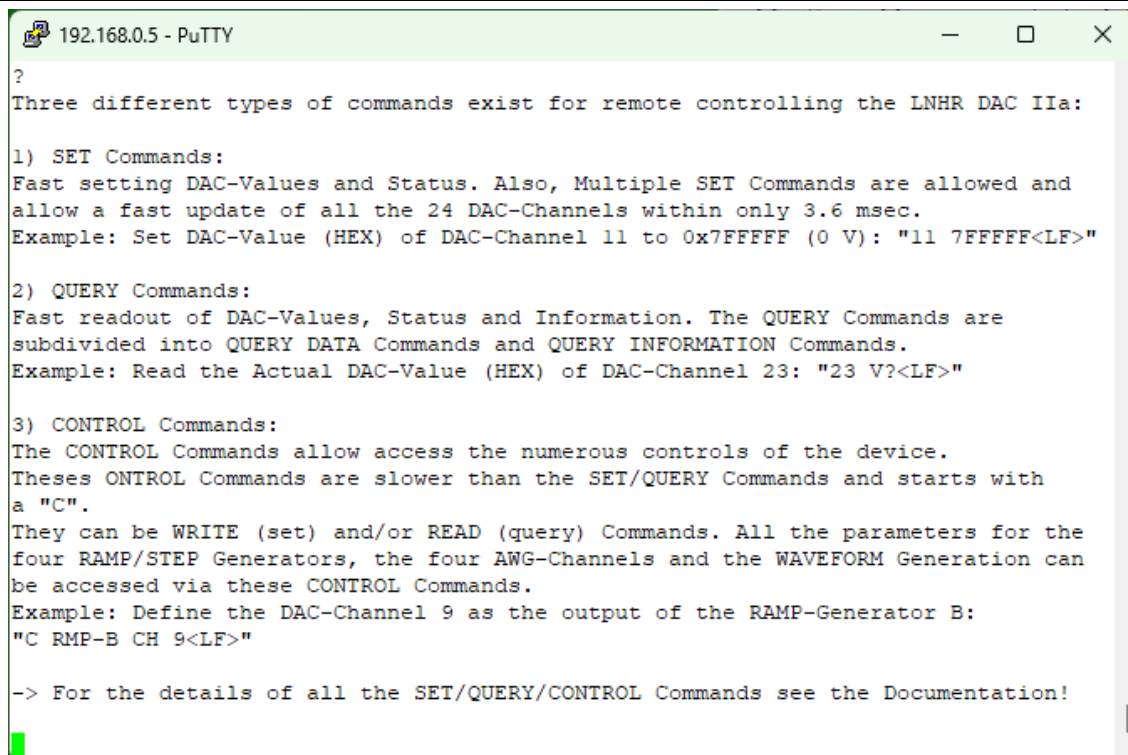
7.2 QUERY INFORMATION Commands (Read)

Several information, settings and status can be read from the LNHR DAC IIa. All these QUERY INFORMATION Commands must be terminated with a <LF>:

"?"	Shows an overview of the ASCII COMMANDS and QUERIES
"HELP?"	Shows the HELP text (as on local LCD)
"SOFT?"	Shows the software release of the device (as on local LCD)
"HARD?"	Shows information on the used hardware (as on local LCD)
"IDN?"	Shows the device identification with serial number
"HEALTH?"	Shows the health of the device (Temps, CPU-load, power-supplies)
"IP?"	Shows the actual IP-Address and the Subnet-Mask (as on local LCD)
"SERIAL?"	Shows the actual Baud-Rate of the RS-232 port (as on local LCD)
"CONTACT?"	Shows the contact address in case of problems (as on local LCD)

The ASCII-text can be displayed on a standard terminal-program (e.g. PuTTY). Here are some examples of QUERY INFORMATION Commands send via serial port (RS-232) by using the PuTTY terminal-program:

7.2.1 Query "?"



```

192.168.0.5 - PuTTY
?
Three different types of commands exist for remote controlling the LNHR DAC IIa:

1) SET Commands:
Fast setting DAC-Values and Status. Also, Multiple SET Commands are allowed and
allow a fast update of all the 24 DAC-Channels within only 3.6 msec.
Example: Set DAC-Value (HEX) of DAC-Channel 11 to 0x7FFFFFF (0 V): "11 7FFFFFF<LF>"

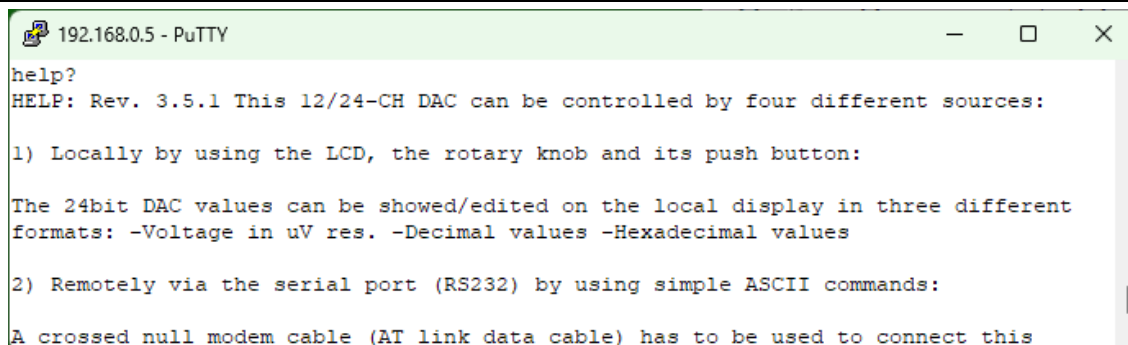
2) QUERY Commands:
Fast readout of DAC-Values, Status and Information. The QUERY Commands are
subdivided into QUERY DATA Commands and QUERY INFORMATION Commands.
Example: Read the Actual DAC-Value (HEX) of DAC-Channel 23: "23 V?<LF>"

3) CONTROL Commands:
The CONTROL Commands allow access the numerous controls of the device.
Theses ONTROL Commands are slower than the SET/QUERY Commands and starts with
a "C".
They can be WRITE (set) and/or READ (query) Commands. All the parameters for the
four RAMP/STEP Generators, the four AWG-Channels and the WAVEFORM Generation can
be accessed via these CONTROL Commands.
Example: Define the DAC-Channel 9 as the output of the RAMP-Generator B:
"C RMP-B CH 9<LF>"

-> For the details of all the SET/QUERY/CONTROL Commands see the Documentation!

```

7.2.2 Query "HELP?"



```

192.168.0.5 - PuTTY
help?
HELP: Rev. 3.5.1 This 12/24-CH DAC can be controlled by four different sources:

1) Locally by using the LCD, the rotary knob and its push button:

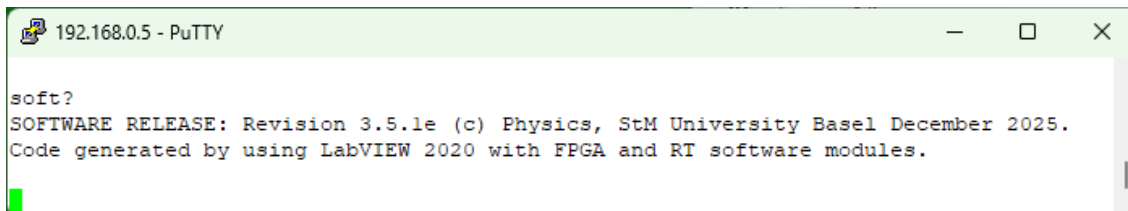
The 24bit DAC values can be showed/edited on the local display in three different
formats: -Voltage in uV res. -Decimal values -Hexadecimal values

2) Remotely via the serial port (RS232) by using simple ASCII commands:

A crossed null modem cable (AT link data cable) has to be used to connect this

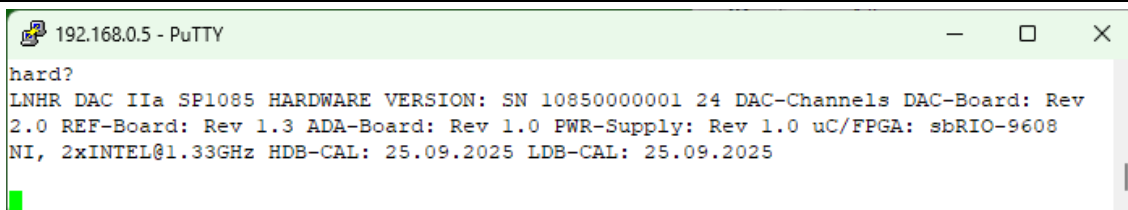
```

7.2.3 Query “SOFT?”



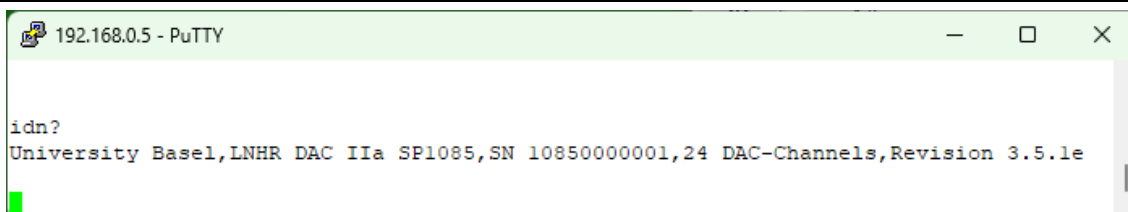
```
192.168.0.5 - PuTTY
soft?
SOFTWARE RELEASE: Revision 3.5.1e (c) Physics, StM University Basel December 2025.
Code generated by using LabVIEW 2020 with FPGA and RT software modules.
```

7.2.4 Query “HARD?”



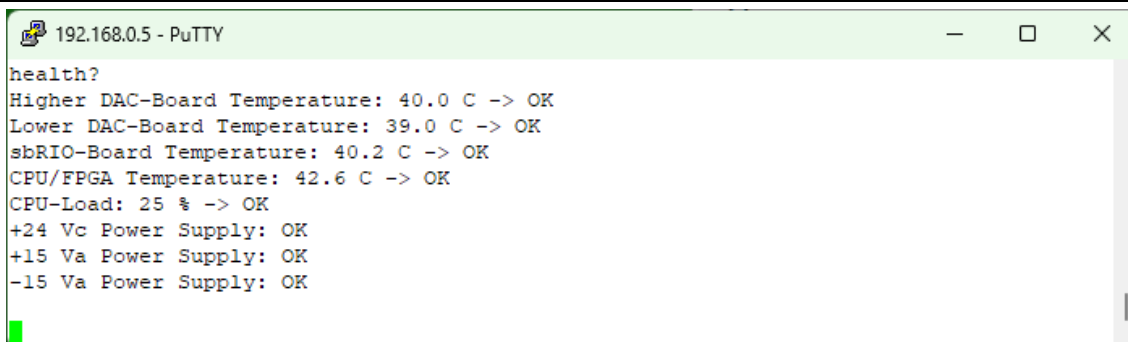
```
192.168.0.5 - PuTTY
hard?
LNHR DAC IIa SP1085 HARDWARE VERSION: SN 10850000001 24 DAC-Channels DAC-Board: Rev
2.0 REF-Board: Rev 1.3 ADA-Board: Rev 1.0 PWR-Supply: Rev 1.0 uC/FPGA: sbRIO-9608
NI, 2xINTEL@1.33GHz HDB-CAL: 25.09.2025 LDB-CAL: 25.09.2025
```

7.2.5 Query “IDN?”



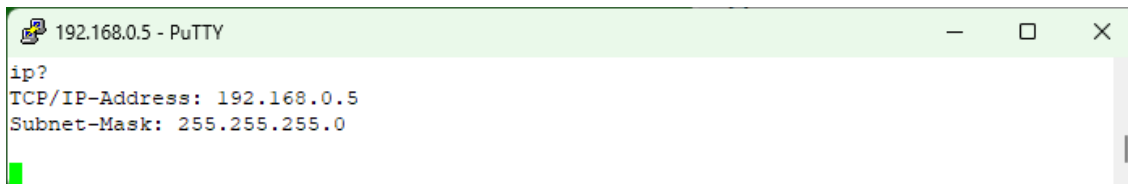
```
192.168.0.5 - PuTTY
idn?
University Basel, LNHR DAC IIa SP1085, SN 10850000001, 24 DAC-Channels, Revision 3.5.1e
```

7.2.6 Query “HEALTH?”



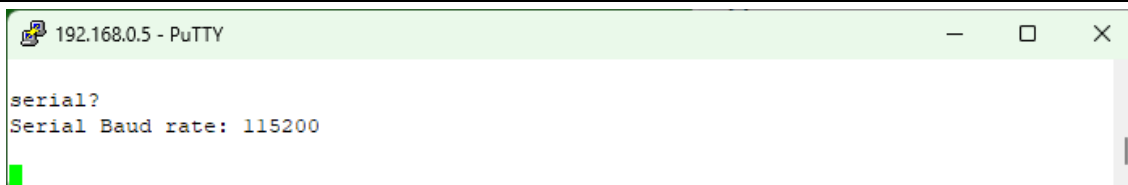
```
192.168.0.5 - PuTTY
health?
Higher DAC-Board Temperature: 40.0 C -> OK
Lower DAC-Board Temperature: 39.0 C -> OK
sbRIO-Board Temperature: 40.2 C -> OK
CPU/FPGA Temperature: 42.6 C -> OK
CPU-Load: 25 % -> OK
+24 Vc Power Supply: OK
+15 Va Power Supply: OK
-15 Va Power Supply: OK
```

7.2.7 Query “IP?”



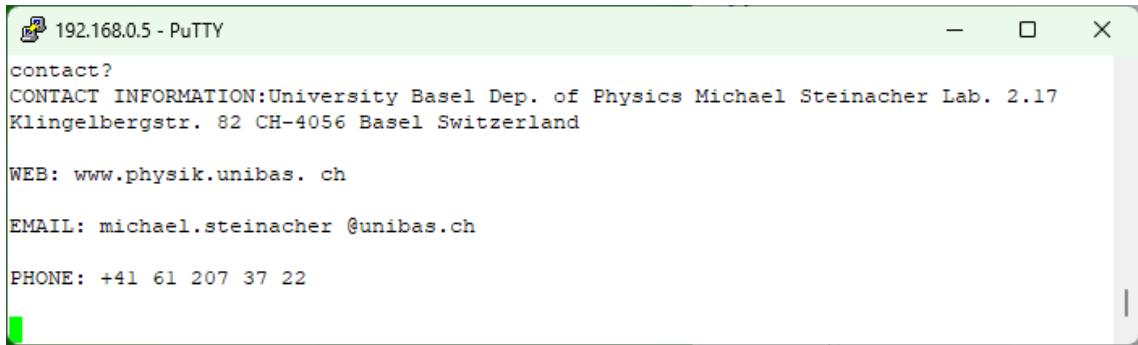
```
192.168.0.5 - PuTTY
ip?
TCP/IP-Address: 192.168.0.5
Subnet-Mask: 255.255.255.0
```

7.2.8 Query “SERIAL?”



```
192.168.0.5 - PuTTY
serial?
Serial Baud rate: 115200
```

7.2.9 Query “CONTACT?”



```
192.168.0.5 - PuTTY
contact?
CONTACT INFORMATION:University Basel Dep. of Physics Michael Steinacher Lab. 2.17
Klingelbergstr. 82 CH-4056 Basel Switzerland

WEB: www.physik.unibas. ch

EMAIL: michael.steinacher @unibas.ch

PHONE: +41 61 207 37 22
```

8 CONTROL Commands (Read / Write)

With the CONTROL Commands the DAC Update-Modes and the numerous controls for the four RAMP/STEP-Generators (RMP-A/B/C/D) and the six AWG-Channels (AWG-A/B/E/C/D/F) can be remotely accessed. Many of the CONTROLS are Boolean (0/1), some integer number (e.g. DAC-Channel) or floating-point value (e.g. Start- and Stop-Voltage).

All CONTROL Commands start with a "C" (to identify a control command), follow by the ASCII CONTROL Command string. Between the "C" and the Command string a <SPACE> makes the separation. The Command must be terminated by a <LF>. Only single CONTROL Commands can be released.

Some of the CONTROLS are write-only, some are read-only but most allow read- and write access. A read/query CONTROL Command always ends with a question-mark ("?") followed by a <LF>. The device responses with a question-mark ("?"), followed by a <CR+LF>, if a read/query CONTROL Command is unknown.

The CONTROL Commands allow accessing to the LNHR DAC IIa at a moderate speed; commands can be repeated at a maximum rate of around 100 Hz (10 msec). Nevertheless, it is mandatory that the response (error-code or data) of the device is read and interpreted before a next CONTROL Command is transmitted. This handshaking allows a stable and reliable data transfer between the host computer and the device.

When a write (set) CONTROL Command is transmitted, the device releases an "Error-Code" with the following meaning:

- "0" = No error (normal)
- "1" = Invalid DAC-Channel
- "2" = Invalid Parameter
- "4" = Mistyped
- "5" = Writing not allowed

CAUTION: It is recommended to wait around 200 msec after sending a CONTROL Command so that device can properly update its internal parameters and synchronized them also with the DAC Commander application.

The detailed description of all CONTROL Commands is grouped in the following six chapters from 7.1 to 7.6:

- 8.1 DAC Update-Mode & Synchronization: Control the DAC synchronization**
- 8.2 RAMP/STEP-Generator:** Control the four RAMP- and STEP-Generators.
- 8.3 2D-Scan:** Control the parameters for 2D-Scans.
- 8.4 AWG:** Control the six AWG-Generators.
- 8.5 Standard Waveform Generation (SWG):** Control the parameters for the generation of standard waveforms which can be written to the six Wave-Memories (WAV-A/B/E/C/D/F).
- 8.6 Wave:** Control the seven Wave-Memories (WAV-A/B/E/C/D/F/S) and write them to the six AWG-Memories (AWG-A/B/E/C/D/F)

8.1 DAC Update-Mode & Synchronization CONTROL Commands

The Update-Mode – Instantly or Synchronous – of the two DAC-Boards can be selected or readout. In the Instantly-Mode the DAC is transparent and the written DAC-Values are immediately converted to an output voltage; this corresponds to the DAC Mode “DAC”.

In the Synchronous-Mode the written DAC-Values are only registered and not converted to an output voltage. All the DACs are parallel updated by a SYNC-Command (internal) or by an external signal. This allows a synchronous update of all the 12 DAC-Channels on one DAC-Board (Lower or Higher) or all the 24 DAC-Channels on both DAC-Boards (Lower and Higher) in parallel.

8.1.1 DAC Update-Mode (Read / Write)

Read or write the Update-Mode of the two DAC-Boards (Lower or Higher). The returned integer number defines the actual Update-Mode:

0=Instantly/1=Synchronous

Read:

```
"C <SPACE> DAC-Board[UM-L,UM-H]?<LF>"
  → "Update-Mode[0,1]<CR+LF>"
```

Examples :

Read Update-Mode of Lower DAC-Board (Instantly): "C UM-L?<LF>"
 → "0<CR/LF>"

Read Update-Mode of Higher DAC-Board (Synchronous): "C UM-H?<LF>"
 → "1<CR/LF>"

Write:

```
"C <SPACE> DAC-Board[UM-L,UM-H] <SPACE> Update-Mode[0/1]<LF>"
```

Examples:

Write Update-Mode Instantly to the Lower DAC-Board: "C UM-L 0<LF>"

Write Update-Mode Synchronous to the Higher DAC-Board: "C UM-H 1<LF>"

CAUTION: Wait at least 200 msec from setting the DAC Update-Mode to Synchronous, before the first SYNC Command is send!

8.1.2 Synchronous DAC-Update (Write only)

Make a Synchronous DAC-Update of all the 12 DAC-Channels on one DAC-Board (Lower=SYNC-L or Higher=SYNC-H) or on both in parallel (Lower and Higher=SYNC-LH). After setting these controls, they are reset internally.

```
"C <SPACE> Board-SYNC[SYNC-L/SYNC-H/SYNC-LH]<LF>"
```

Examples:

Synchronous DAC-Update on Lower DAC-Board only: "C SYNC-L<LF>"

Synchronous DAC-Update on Higher DAC-Board only: "C SYNC-H<LF>"

Synchronous DAC-Update on Lower and Higher DAC-Boards: "C SYNC-LH<LF>"

8.2 RAMP/STEP-Generator CONTROL Commands

The four RAMP/STEP-Generators (RMP-A/B/C/D) can be programmed and controlled by the following CONTROL Commands.

8.2.1 RAMP Start/Hold/Stop (Write only)

Control the mode of the four RAMP/STEP-Generators (RMP-A/B/C/D) by the Boolean controls Start, Stop and Hold. After setting these controls, they are reset internally. RMP-ALL makes synchronous access to all the four RAMP/STEP-Generators. ALL RAMP/STEP-Generators can only be started when all are in Idle- or Hold-Mode.

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D/ALL] START/HOLD/STOP<LF>"
```

Examples:

Start RAMP/STEP-Generator A: "C RMP-A START<LF>"

Hold RAMP/STEP-Generator B: "C RMP-B HOLD<LF>"

Stop RAMP/STEP-Generator C: "C RMP-C STOP<LF>"

Stop ALL RAMP/STEP-Generator: "C RMP-ALL STOP<LF>"

8.2.2 RAMP State (Read only)

Readout the State (Idle/Ramp_UP/Ramp_DOWN/Hold) of the four RAMP/STEP-Generators (RMP-A/B/C/D). The returned integer number defines the actual state: 0=Idle/1=Ramp_UP/2=Ramp_DOWN/3=Hold

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> S?<LF>"
  → "State[0/1/2/3]<CR+LF>"
```

Examples:

Read State of RAMP/STEP-Generator A: "C RMP-A S?<LF>"
 → "0<CR/LF>" (The RMP-A is in Idle State)

Read State of RAMP/STEP-Generator D: "C RMP-D S?<LF>"
 → "2<CR/LF>" (The RMP-D is in Ramp_DOWN State)

8.2.3 RAMP Cycles-Done (Read only)

Readout the Cycles-Done of the four RAMP/STEP-Generators (RMP-A/B/C/D). The returned integer number represents the completed RAMP Cycles since the Start; it is in a range from 0 to 4E9. The number is valid in all of the RAMP States. When the RAMP is stopped (state Idle) the RAMP Cycles-Done shows the last value; when the RAMP/STEP-Generator is started, the corresponding Cycles-Done is reset to zero (0). After power-up all the Cycles-Done counters are zero (0).

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> CD?<LF>"
  → "Cycles-Done[0...4E9]<CR+LF>"
```

Examples:

Read Cycles-Done of RAMP/STEP-Generator A: "C RMP-A CD?<LF>"
 → "145<CR/LF>"

Read Cycles-Done of RAMP/STEP-Generator D: "C RMP-D CD?<LF>"
 → "0<CR/LF>"

8.2.4 RAMP Steps-Done (Read only)

Readout the Steps-Done of the four RAMP/STEP-Generators (RMP-A/B/C/D). The returned integer number represents the Steps-Done since the Start. It is in a range from 0 to 4E9. In the states Ramp_UP, Ramp_DOWN and Hold the number indicates the Steps-Done since the RAMP/STEP-Generator was started. When the RAMP/STEP-Generator is stopped (state Idle) the Steps-Done counter is reset to zero (0). After power-up all the Cycles-Done counters are zero (0).

Note: If RAMP Shape is /UP-DOWN\ (Triangle function) the STEPs-Done counter decrements to zero after reaching the Peak Voltage. It counts up during Ramp_UP state and counts down to zero during Ramp_DOWN state.

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> SD?<LF>"
  → "Cycles-Done[0...4E9]<CR+LF>"
```

Examples:

```
Read Steps-Done of RAMP/STEP-Generator A: "C RMP-A SD?<LF>"
  → "681<CR/LF>" (681 Steps since the last start)
```

```
Read Steps-Done of RAMP/STEP-Generator C: "C RMP-C SD?<LF>"
  → "0<CR/LF>" (No Steps Done – in Idle Mode)
```

8.2.5 RAMP Step-Size Voltage (Read only)

Readout the Step-Size Voltage [V/STEP] of the four RAMP/STEP-Generators (RMP-A/B/C/D), calculated from the corresponding RAMP/STEP-Parameters. It is also depending on the selected shape of the Ramping/Stepping function ("UP-ONLY", "UP-DOWN"). The returned Step-Size Voltage is a floating-point/exponential (E) number with a period (point) as the decimal point.

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> SSV?<LF>"
  → "Step-Size Voltage[±10.000000]<CR+LF>"
```

Examples:

```
Read Step-Size Voltage of RAMP/STEP-Generator A: "C RMP-A SSV?<LF>"
  → "1E-3<CR/LF>"
```

```
Read Step-Size Voltage of RAMP/STEP-Generator D: "C RMP-D SSV?<LF>"
  → "1E-5<CR/LF>"
```

8.2.6 RAMP Steps per Cycle (Read only)

Readout the number of Steps per Cycle, calculated from the RAMP/STEP-Parameters. The returned integer value is the number of Steps performed within one Cycle. It is calculated from the RAMP-Time divided by 5 msec, which is the fixed RAMP update rate.

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> ST?<LF>"
  → "Steps_per_Cycle[10... 200'000'000]<CR+LF>"
```

Examples:

```
Read Steps per Cycle of RAMP/STEP-Generator A: "C RMP-A ST?<LF>"
  → "2000<CR/LF>"
```

```
Read Steps per Cycle of RAMP/STEP-Generator B: "C RMP-B ST?<LF>"
  → "112345<CR/LF>"
```

8.2.7 RAMP DAC-Channel AVAILABLE (Read only)

Readout if the selected DAC-Channel of the RAMP/STEP-Generator (RMP-A/B/C/D) is available (not used by other RAMP- or AWG-Channels). The returned integer number gives the availability:

0=Not Available/1=Available

Note: A running RAMP/STEP-Generator reads always “Not Available (0)” for its DAC-Channel.

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> AVA?<LF>"
 → "Available[0/1]<CR+LF>"

Examples:

Read availability of DAC-Channel of RAMP/STEP-Generator C: "C RMP-C AVA?<LF>"
 → "1<CR/LF>" (means Available)

Read availability of DAC-Channel of RAMP/STEP-Generator D: "C RMP-D AVA?<LF>"
 → "0<CR/LF>" (means Not Available)

8.2.8 RAMP Selected DAC-CHANNEL (Read / Write)

Read or write the Selected DAC-Channel for the RAMP/STEP-Generator (RMP-A/B/C/D). The DAC-Channel can be in the range from 1 to 24. After writing the Selected DAC-Channel, its availability can be checked – see above.

Read:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> CH?<LF>"
 → "DAC-Channel[1...24]<CR+LF>"

Example:

Read Selected DAC-Channel of RAMP/STEP-Generator B: "C RMP-B CH?<LF>"
 → "2<CR/LF>"

Write:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> CH <SPACE>
 DAC-Channel[1...24]<LF>"

Example:

Write Selected DAC-Channel 8 for RAMP/STEP-Generator D Output: "C RMP-D CH 8<LF>"

8.2.9 RAMP Start Voltage (Read / Write)

Read or write the Start Voltages of the RAMP/STEP-Generators (RMP-A/B/C/D). The Start Voltage is a floating-point number in the range between -10.000000 V and +10.000000 V. The decimal point must be a period (point).

Read:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STAV?<LF>"
 → "Voltage[±10.000000]<CR+LF>"

Example:

Read Start Voltage of RAMP/STEP-Generator A: "C RMP-A STAV? <LF>"
 → "-1.123456<CR/LF>"

Write:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STAV <SPACE> Voltage[±10.000000]<LF>"

Example:

Write Start Voltage of RAMP/STEP-Generator D to -3.232 V: "C RMP-D STAV -3.232<LF>"

8.2.10 RAMP Stop/Peak Voltage (Read / Write)

Read or write the Stop/Peak Voltages of the RAMP/STEP-Generators (RMP-A/B/C/D). The Stop/Peak Voltage is a floating-point number in the range between -10.000000 V and +10.000000 V. The decimal point must be a period (point). If the RAMP Shape is UP-ONLY (Sawtooth function) this value is the Stop Voltage. If the RAMP Shape is UP and DOWN (Triangle function) this value is the Peak Voltage, since the Triangle function returns to the Start Voltage.

Read:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STOV?<LF>"
→ "Voltage[±10.000000]<CR+LF>"

Example:

Read Stop Voltage of RAMP/STEP-Generator A: "C RMP-A STOV?<LF>"
→ "3.123456<CR+LF>"

Write:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STOV <SPACE> Voltage[±10.000000]<LF>"

Example:

Write Stop Voltage of RAMP/STEP-Generator D to -1.214 V: "C RMP-D STOV -1.214<LF>"

8.2.11 RAMP Time (Read / Write)

Read or write the RAMP Times of the RAMP/STEP-Generators (RMP-A/B/C/D). The RAMP Time is a floating-point number in the range between 0.05 second and 1E6 seconds (equal to 277.7 hours). The decimal point must be a period (point). The inherent resolution is given by the RAMP/STEP-Generator cycle of 5 msec (0.005 second).

Read:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> RT?<LF>"
→ "Time[0.05...1E6]<CR+LF>"

Example:

Read RAMP Time of RAMP/STEP-Generator B: "C RMP-B RT?<LF>"
→ "120.885<CR+LF>"

Write:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> RT <SPACE> Time[0.05...1E6]<LF>"

Example:

Write RAMP Time of 221.3 seconds to RAMP/STEP-Generator D: "C RMP-D RT 221.3<LF>"

8.2.12 RAMP Shape (Read / Write)

Read or write the RAMP Shape of the RAMP/STEP-Generators (RMP-A/B/C/D). Two different Shapes of the Ramping/Stepping function can be written or readout:
0=UP-ONLY (Sawtooth function) /1=UP and DOWN (Triangle function)

Read:

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> RS?<LF>"
  → "Shape[0/1]<CR+LF>"
```

Example:

Read RAMP Shape of RAMP/STEP-Generator D: "C RMP-D RS?<LF>"
→"0<CR/LF>" (means UP-ONLY)

Write:

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> RS <SPACE>
Shape[0/1]<LF>"
```

Example:

Write RAMP Shape UP and DOWN to RAMP/STEP-Generator A: "C RMP-A RS 1<LF>"

8.2.13 RAMP Cycles-Set (Read / Write)

Read or write the number of RAMP Cycles-Set (0...4E9) of the four RAMP/STEP-Generators (RMP-A/B/C/D). The integer number represents the number of RAMP Cycles-Set until the RAMP/STEP-Generator gets stopped. If the number of RAMP Cycles-Set is zero (0), an infinite number of RAMP Cycles are done; the RAMP/STEP-Generator runs until it is stopped by the user.

Read:

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> CS?<LF>"
  → "Cycles-Set[0...4E9]<CR+LF>"
```

Example:

Read Cycles-Set of RAMP/STEP-Generator B: "C RMP-B CS?<LF>"
→"25<CR/LF>"

Write:

```
"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> CS <SPACE>
Cycles-Set[0...4E9]<LF>"
```

Example:

Write Cycles-Set of 342 to RAMP/STEP-Generator D: "C RMP-D CS 342<LF>"

8.2.14 RAMP/STEP-Selection (Read / Write)

Read or write the RAMP/STEP-Selection of the four RAMP/STEP-Generators (RMP-A/B/C/D). The RAMP/STEP-Generators can be switched between RAMP function and STEP function. The normal RAMP function periodically updates the DAC-Voltage each 5 msec. The STEP function updates the DAC-Voltage when the AWG has stopped (single cycle); this used for 2D-Scans. The two different functions (RAMP/STEP) can be written or readout:

0=RAMP function/1=STEP function

Read:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STEP?<LF>"
→ "RAMP/STEP[0/1]<CR+LF>"

Examples:

Read the RAMP/STEP of RAMP/STEP-Generator B: "C RMP-B STEP?<LF>"

→"0<CR/LF>" (means RAMP function)

Read the RAMP/STEP of RAMP/STEP-Generator D: "C RMP-D STEP?<LF>"

→"1<CR/LF>" (means STEP function)

Write:

"C <SPACE> RAMP/STEP-Generator[RMP-A/B/C/D] <SPACE> STEP <SPACE>
RAMP/STEP[0/1]<LF>"

Examples:

Set RAMP/STEP-Generator C to STEP function: "C RMP-C STEP 1<LF>"

Set RAMP/STEP-Generator A to RAMP function: "C RMP-A STEP 0<LF>"

8.3 2D-Scan CONTROL Commands

These CONTROL Commands are used to program and readout the parameters for a 2D-Scan. These parameters are only active when the RAMP/STEP-Generator is defined as a STEP-Generator (see above).

The 2D-Scan combines the STEP-Generator (RMP-A/B/C/D) with the corresponding AWG-Function (AWG-A/B/C/D). For this purpose, the AWG performs a fast Ramp/Triangle function (y-axis) while the linear STEP-Generator (x-axis) gets updated after the AWG has stopped. The number of AWG Cycles must be set to one (1). Further it is needed to set also the RAMP Cycles to one (1) and adapt the RAMP Time to reach the desired number of STEPs in the x-axis. The number of STEPs per cycle can be varied from 10 (RAMP Time = 0.05 sec) up to 200'000'000 (RAMP Time = 1E6 sec). Thus, the number of x-lines of a 2D-Scan can be set from 10 to 2E8.

8.3.1 Normal-Start / Auto-Start AWG (Read / Write)

Read or write the Boolean parameter Normal-Start / Auto-Start AWG. If Auto-Start AWG is selected (1) the AWG gets automatically restarted after the STEP-Generator has been updated. A minimum time delay of 5 msec is implemented from the update of the STEP-Generator to the restart of the AWG.

If Auto-Start AWG is deselected (0) the AWG starts normally by an external TTL-Trigger or via a CONTROL Command. The two different modes (Normal-Start AWG or Auto-Start AWG) can be written or readout:

0=Normal-Start AWG/1=Auto-Start AWG

CAUTION: For a reliable auto-start, the AWG Duration/Period [sec] must be at least 6 msec.

Read:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> AS?<LF>"
  → "Normal/Auto-Start AWG[0/1]<CR+LF>"
```

Examples:

Read the Normal/Auto-Start of AWG-C by STEP-Generator C: "C AWG-C AS?<LF>"
 → "0<CR/LF>" (means Normal-Start AWG)

Read the Normal/Auto-Start of AWG-D by STEP-Generator D: "C AWG-D AS?<LF>"
 → "1<CR/LF>" (means Auto-Start AWG)

Write:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> AS <SPACE>
Normal/Auto-Start AWG[0/1]<LF>"
```

CAUTION: It is not allowed to alter this parameter during a running 2D-Scan!

Examples:

Set Auto-Start AWG-C by STEP-Generator C: "C AWG-C AS 1<LF>"

Set Normal-Start AWG-A: "C AWG-A AS 0<LF>"

8.3.2 Keep / Reload AWG MEM (Read / Write)

Read or write the Boolean parameter Keep/Reload AWG MEM. If Reload AWG MEM is selected (1) the AWG-Memory (AWG-A/B/C/D) is reloaded from the corresponding

Wave-Memory (WAV-A/B/C/D) before it gets restarted. This option is slower, but has to be selected since the Polynomial (POLY-A/B/C/D) must be applied to perform an adaptive 2D-Scan.

If Keep AWG MEM is selected (0) the predefined AWG-Memory is used for a next scan-line, which allows a faster 2D-Scan, but without adaptation. These two different behaviors (Keep AWG MEM or Reload AWG MEM) can be written or readout:

0=Keep AWG MEM/1=Reload AWG MEM

Read:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> RLD?<LF>"
  → "Keep/Reload AWG MEM[0/1]<CR+LF>"
```

Examples:

Read the Keep/Reload AWG MEM of AWG-C: "C AWG-C RLD?<LF>"
 → "0<CR/LF>" (means Keep AWG MEM)

Read the Keep/Reload AWG MEM of AWG-D: "C AWG-D RLD?<LF>"
 → "1<CR/LF>" (means Reload AWG MEM)

Write:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> RLD <SPACE>
Keep/Reload AWG MEM[0/1]<LF>"
```

Examples:

Set AWG-B to Keep AWG MEM: "C AWG-B RLD 0<LF>"

Set AWG-D to Reload AWG MEM: "C AWG-D RLD 1<LF>"

8.3.3 Skip / Apply Polynomial (Read / Write)

Read or write the Boolean parameter Skip/Apply Polynomial. If Apply Polynomial is selected (1) the Polynomial (POLY-A/B/C/D) is applied when the AWG-Memory (AWG-A/B/C/D) is reloaded from the corresponding Wave-Memory (WAV-A/B/C/D); this is essential for an adaptive 2D-Scan. The Polynomial (POLY-A/B/C/D) can be updated fast by using the SET POLY Command or by using the parameter "Adaptive Shift-Voltage" – see description below.

If Skip Polynomial is selected the Polynomial (POLY-A/B/C/D) isn't applied when the predefined Wave-Memory (WAV-A/B/C/D) is written to the AWG-Memory (AWG-A/B/C/D). These two different behaviors (Skip Polynomial or Apply Polynomial) can be written or readout:

0=Skip Polynomial/1=Apply Polynomial

Read:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> AP?<LF>"
  → "Skip/Apply Polynomial[0/1]<CR+LF>"
```

Examples:

Read the Skip/Apply Polynomial of AWG-A: "C AWG-A AP?<LF>"
 → "1<CR/LF>" (means Apply Polynomial)

Read the Skip/Apply Polynomial of AWG-B: "C AWG-B AP?<LF>"
 → "0<CR/LF>" (means Skip Polynomial)

Write:

```
"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> AP <SPACE>
```

Skip/Apply Polynomial[0/1]<LF>”

Examples:

Set AWG-D to Apply Polynomial: “C AWG-D AP 1<LF>”

Set AWG-C to Skip Polynomial: “C AWG-C AP 0<LF>”

8.3.4 Adaptive Shift-Voltage (Read / Write)

Read or write the Adaptive Shift-Voltage per Step of the STEP-Generators (RMP-A/B/C/D). A simple adaptive 2D-Scan (with linear y-adaption) can be performed by this parameter. This Shift-Voltage gets applied to the AWG function (y-Axis) after each step of the STEP-Generator (x-Axis). This is automatically done by linearly modifying the polynomial coefficient a_0 (constant) after each step. This coefficient a_0 is calculated by multiplying the Adaptive Shift-Voltage by the RAMP Cycles-Done. If the Adaptive Shift-Voltage is zero (0) the polynomial will not be changed.

Note: Since it is allowed to modify this Adaptive Shift-Voltage while a 2D-Scan is running, also nonlinear adaptations can be easily implemented.

The Adaptive Shift-Voltage is a floating-point number in the range between -10.000000 V and +10.000000 V per Step. The decimal point must be a period (point).

Read:

"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> SHIV?<LF>"

→ "Adaptive Shift-Voltage[±10.000000]<CR+LF>"

Examples:

Read the Adaptive Shift-Voltage of AWG-A: “C AWG-A SHIV?<LF>”

→ “0.01<CR/LF>” (means an Adaptive Shift Voltage of 10 mV)

Read the Adaptive Shift-Voltage of AWG-C: “C AWG-C SHIV?<LF>”

→ “0<CR/LF>” (0 means no adaption via the polynomial coefficient a_0)

Write:

"C <SPACE> AWG[AWG-A/B/C/D] <SPACE> SHIV <SPACE> Voltage[±10.000000]<LF>"

Examples:

Set Adaptive Shift-Voltage (per Step) for AWG-A to 12 mV: “C AWG-A SHIV 12E-3<LF>”

Set Adaptive Shift-Voltage (per Step) for AWG-B to -9.8 mV: “C AWG-B SHIV -9.8E-3<LF>”

8.4 AWG CONTROL Commands

The six Arbitrary Waveform Generators (AWG-A/B/E/C/D/F) can be programmed and controlled by the following CONTROL Commands.

8.4.1 AWG Normal / AWG Only (Read / Write)

Read or write the Boolean parameter AWG Normal/AWG Only, related to the Lower DAC Board (AWG-A/B/E) or to the Higher DAC-Board (AWG-C/D/F). If AWG Only is selected (1), all the other DAC-CHANNELS on this DAC-Board get blocked (---) and only the AWG-Channels are active. This results in lower time-jitter for these AWG-Channels.

If AWG Normal is selected (0) the other DAC-CHANNELS on the corresponding DAC-Board are free to be used for normal DAC operation (DAC) or as RAMP/STEP-Generator (RMP). These two different options (AWG Normal or AWG Only) can be written or readout:
0=AWG Normal/1=AWG Only

Read:

```
"C <SPACE> AWG-Group[AWG-AB(E)/CD(F)] <SPACE> ONLY?<LF>"
  → "AWG Normal/Only[0/1]<CR+LF>"
```

Examples:

Read the AWG Normal/Only of the Lower DAC-Board (ABE): "C AWG-ABE ONLY?<LF>"
→ "0<CR/LF>" (means AWG Normal)

Read the AWG Normal/Only of the Higher DAC-Board (CDF): "C AWG-CDF ONLY?<LF>"
→ "1<CR/LF>" (means AWG Only)

Write:

```
"C <SPACE> AWG-Group[AWG-AB(E)/CD(F)] <SPACE> ONLY <SPACE>
AWG Normal/Only[0/1]<LF>"
```

Examples:

Set AWG Only on the Lower DAC-Board (ABE): "C AWG-ABE ONLY 1<LF>"

Set AWG Normal on the Higher DAC-Board (CDF): "C AWG-CDF ONLY 0<LF>"

Note: For reasons of backward compatibility, the AWG-Groups can be named as AB or ABE (Lower DAC-Board) and as CD or CDF (Higher DAC-Board).

8.4.2 AWG Start / Stop (Write only)

Control the mode of the six AWGs (AWG-A/B/E/C/D/F) by the two Boolean controls Start and Stop. After setting these controls, they are reset internally. AWG-AB(E) allows synchronous access to the two (three) AWGs on the Lower DAC-Board and AWG-CD(F) to the two (three) AWGs on the Higher DAC-Board. AWG-ALL makes synchronous access to all the six AWGs. A synchronous start of multiple AWGs (AB(E)/CD(F)/ALL) can only be performed when all these AWGs are in Idle-Mode.

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F/AB(E)/CD(F)/ALL] START/STOP<LF>"
```

Examples:

Start AWG A: "C AWG-A START<LF>"

Stop AWG B: "C AWG-B STOP<LF>"

Stop AWG C and D (Higher DAC-Board): "C AWG-CD STOP<LF>"

Start AWG A, B and E (Lower DAC-Board): "C AWG-ABE START<LF>"

8.4.3 AWG State (Read only)

Readout the State (Idle/Running) of the six AWGs (AWG-A/B/E/C/D/F). The returned integer number defines the actual state:

0=Idle/1=Running

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> S?<LF>"
  → "State[0/1]<CR+LF>"
```

Examples:

```
Read State of AWG-A (Idle): "C AWG-A S?<LF>"
  → "0<CR+LF>"
```

```
Read State of AWG-F (Running): "C AWG-F S?<LF>"
  → "1<CR+LF>"
```

8.4.4 AWG Cycles-Done (Read only)

Readout the AWG Cycles-Done of the six AWGs (AWG-A/B/E/C/D/F). The returned integer number represents the completed AWG Cycles since the Start. It is in a range from 0 to 4E9. When the AWG is stopped (state Idle) the AWG Cycles-Done shows the last value. When the AWG is started it is reset to zero (0). After power-up all the AWG Cycles-Done counters are zero (0).

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> CD?<LF>"
  → "Cycles-Done[0...4E9]<CR+LF>"
```

Examples:

```
Read Cycles-Done of AWG-A: "C AWG-A CD?<LF>"
  → "34617<CR+LF>"
```

```
Read Cycles-Done of AWG-E: "C AWG-E CD?<LF>"
  → "0<CR+LF>"
```

8.4.5 AWG Duration/Period (Read only)

Readout the Duration/Period of a complete AWG-Cycle (AWG Memory_Size * AWG Clock-Period) of the six AWGs (AWG-A/B/E/C/D/F). The unit of the Duration/Period is second (sec). The minimum AWG Memory Size is 2 and the minimum AWG Clock-Period is 10E-6 sec (10 µsec), resulting in a minimum Duration/Period is 20E-6 sec (20 µsec). The maximum Duration/Period of 2.6E8 seconds (8.2 years) is given by the maximum AWG Memory Size of 65'000 and the maximum AWG Clock-Period of 4'000 sec (equal to 4E9 µsec).

The Duration/Period is a floating-point/exponential (E) number with a period (point) as decimal separator.

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> DP?<LF>"
  → "Duration/Period[20E-6...2.6E8]<CR+LF>"
```

Examples:

```
Read Duration/Period of AWG-A: "C AWG-A DP?<LF>"
  → "20E-6<CR+LF>"
```

```
Read Duration/Period of AWG-F: "C AWG-F DP?<LF>"
  → "5.513E-3<CR+LF>"
```

8.4.6 AWG DAC-Channel AVAILABLE (Read only)

Readout if the selected DAC-Channel of the AWG (AWG-A/B/E/C/D/F) is available (not used by other AWG- or RAMP-Channels). The returned integer number gives the availability:

0=Not Available/1=Available

Note: A running AWG reads always “Not Available (0)” on its DAC-Channel.

"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> AVA?<LF>"
 → "Available[0/1]<CR+LF>"

Examples:

Read availability of DAC-Channel of AWG-C: "C AWG-C AVA?<LF>"
 → "1<CR/LF>" (means Available)

Read availability of DAC-Channel of AWG-E: "C AWG-E AVA?<LF>"
 → "0<CR/LF>" (means Not Available)

8.4.7 AWG Selected DAC-CHANNEL (Read / Write)

Read or write the Selected DAC-Channel for the AWG (AWG-A/B/E/C/D/F). Since the AWG-A, AWG-B and the AWG-E are running on the Lower DAC-Board, their DAC-Channels can only be in the range from 1 to 12. The AWG-C, AWG-D and the AWG-E are running on the Higher DAC-Board and therefore the DAC-Channels are restricted to the range from 13 to 24.

After writing the Selected DAC-Channel, its availability can be checked – see above.

Read:

"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> CH?<LF>"
 → "DAC-Channel[1...24]<CR+LF>"

Examples:

Read Selected DAC-Channel of AWG-B: "C AWG-B CH?<LF>"
 → "12<CR/LF>"

Read Selected DAC-Channel of AWG-F: "C AWG-F CH?<LF>"
 → "22<CR/LF>"

Write:

"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> CH <SPACE> DAC-Channel[1...24]<LF>"

Note:

AWG-A, AWG-B and AWG-E only DAC-Channel[1...12]
 AWG-C, AWG-D and AWG-F only DAC-Channel[13...24]

Examples:

Write Selected DAC-Channel 5 for AWG-A Output: "C AWG-A CH 5<LF>"

Write Selected DAC-Channel 24 for AWG-F Output: "C AWG-F CH 24<LF>"

8.4.8 AWG-Memory Size (Read / Write)

Read or write the AWG-Memory Size of the AWG (AWG-A/B/E/C/D/F) which is an integer number in the range from 2 to 65'000. Each point of the AWG-Memory corresponds to a 24-bit DAC-Value.

The AWG streams this AWG-Memory Size number to the DAC when the AWG is started. This is independent of the number of programmed AWG-Memory Addresses. If the user has downloaded an AWG-Waveform consisting of 1'000 points, also the AWG-Memory Size has to be set to 1'000. The AWG-Memory Size is automatically adapted when the Standard Waveform Generator is used.

Read:

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> MS?<LF>"
  → "AWG-Memory Size[2...65'000]<CR+LF>"
```

Example:

Read AWG-Memory Size of AWG-A: "C AWG-A MS?<LF>"
→"1000<CR/LF>"

Write:

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> MS <SPACE>
AWG-Memory Size[2...65'000]<LF>"
```

Example:

Write an AWG-Memory Size of 3'300 to AWG-E: "C AWG-E MS 3300<LF>"

8.4.9 AWG Cycles-Set (Read / Write)

Read or write the number of AWG Cycles-Set (0...4E9) of one of the six AWGs (AWG-A/B/E/C/D/F). The integer number represents the number of AWG Cycles-Set until the AWG gets stopped. If the number of AWG Cycles-Set is zero (0), an infinite number of AWG-Cycles are done; the AWG runs until it is stopped by the user.

Read:

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> CS?<LF>"
  → "Cycles-Set[0...4E9]<CR+LF>"
```

Example:

Read Cycles-Set of AWG-B: "C AWG-B CS?<LF>"
→"2200<CR/LF>"

Write:

```
"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> CS <SPACE> Cycles-Set[0...4E9]<LF>"
```

Example:

Write Cycles-Set of 34 to AWG-F: "C AWG-F CS 34<LF>"

8.4.10 AWG External Trigger Mode (Read / Write)

Read or write the External Trigger Mode of one of the six AWGs (AWG-A/B/E/C/D/F). This sets the behavior of the six digital inputs "Trig In AWG-A/B/E/C/D/F" on the back panel of the device.

The external applied TTL trigger-signals can be programmed to have the following four different functionalities:

- Disabled (0): The trigger-input has no impact.
- START only (1): The AWG starts on a rising edge of the TTL-signal.
- START-STOP (2): The AWG starts on a rising edge and stops on a falling edge.

- SINGLE-STEP (3): The AWG starts on a rising edge of the TTL-signal and the AWG makes a single step for each rising edge. Therefore, the AWG Clock is defined by the external applied TTL trigger-signal from DC up to maximum 100 kHz (PW minimum 2 μ sec).

The integer number of the External Trigger Mode represents these settings:
0=Disabled/1=START only/2=START-STOP/3=SINGLE-STEP

Read:

"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> TM?<LF>"
→ "External Trigger Mode[0/1/2/3]<CR+LF>"

Examples:

Read the External Trigger Mode of AWG-A: "C AWG-A TM?<LF>"
→ "0<CR/LF>" (means Disabled)

Read the External Trigger Mode of AWG-B: "C AWG-B TM?<LF>"
→ "1<CR/LF>" (means START only)

Read the External Trigger Mode of AWG-F: "C AWG-F TM?<LF>"
→ "3<CR/LF>" (means SINGLE-STEP)

Write:

"C <SPACE> AWG[AWG-A/B/E/C/D/F] <SPACE> TM <SPACE>
External Trigger Mode[0/1/2/3]<LF>"

Example:

Write External Trigger Mode START-STOP to AWG-D: "C AWG-D TM 2<LF>"

8.4.11 AWG Clock-Period [μ sec] (Read / Write)

Read or write the Clock-Period [μ sec] (10...4E9) of the six AWGs (AWG-A/B/E/C/D/F). This integer number represents the AWG Clock-Period in μ sec (1E-6 sec). The minimum is 10 μ sec and the maximum 4E9 μ sec which corresponds to 4'000 sec (4E9 μ sec = 1.111 hours). The resolution of the Clock-Period is 1 μ sec.

Note: The new LNHR DAC Ila has its own clock generator for each of the six AWGs (AWG-A/B/E/C/D/F), whereas the old LNHR DAC II only had one clock generator for the Lower DAC-Board (AWG-A/B) and one for the Higher DAC-Board (AWG-C/D). The Lower AWG Clock-Period [μ sec] was common for the AWG-A and the AWG-B while the Higher AWG Clock-Period [μ sec] was common for the AWG-C and the AWG-D.

To ensure backwards compatibility, the old commands affecting both Clock-Periods of AWG-A/B and AWG-C/D remain valid.

Read:

"C <SPACE> AWG[AWG-A/B/E/C/D/F/AB/CD/ABE/CDF] <SPACE> CP?<LF>"
→ "Clock-Period[10...4E9]<CR+LF>"

Note: Reading a Clock-Period of an AWG group (AB/CD/ABE/CDF) returns the value of the lowest AWG. The AWG-AB and AWG-ABE parameters reads the Clock-Period of the AWG-A. The AWG-CD and AWG-CDF parameters reads the Clock-Period of the AWG-C.

Examples:

Read the AWG Clock-Period of AWG-A: "C AWG-A CP?<LF>"
→ "3333<CR/LF>" (means 3'333 μ sec -> 3.333 msec)

Read the AWG Clock-Period of AWG-C and AWG-D: "C AWG-CD CP?<LF>"

→"2321<CR/LF>" (means 2'321 μsec -> 2.321 msec)

(This ensures backward compatibility with the LNHR DAC II, but only the Clock-Period of AWG-C is read.)

Read the AWG Clock-Period of AWG-F: "C AWG-F CP?<LF>"

→"1023<CR/LF>" (means 1023 μsec -> 1.023 msec)

Write:

"C <SPACE> AWG[AWG-A/B/E/C/D/F/AB/CD/ABE/CDF] <SPACE> CP <SPACE>
Clock-Period[10...4E9]<LF>"

Note: Writing a value to an AWG group (AB/CD/ABE/CDF) updates the Clock-Period of the corresponding AWGs. This gives the backwards compatibility with the old commands of the LNHR DAC II.

Examples:

Write an AWG Clock-Period of 121 μsec to AWG-A:

"C AWG-A CP 121<LF>"

Write an AWG Clock-Period of 2355 μsec to the AWG-C and AWG-D:

"C AWG-CD CP 2355<LF>"

(This ensures backward compatibility with the LNHR DAC II.)

8.4.12 AWG 1 MHz Clock Reference ON/OFF (Read / Write)

Read or write the Control-Status (ON/OFF) of the external digital AWG 1 MHz Clock Reference TTL signal (on the two D-SUB connectors on the back-panel). This 1 MHz reference clock is internally used for clocking the AWGs and it can be used to synchronize other devices with the LNHR DAC IIa.

Note: After power-up this 1 MHz reference clock is switched OFF.

It is recommended to switch it ON, only when needed. A running 1 MHz clock increases the power consumption and adds some noise to the other digital TTL output signals on the D-SUB connectors. The returned integer number defines the actual Control-Status of the 1 MHz Clock:

0=OFF/1=ON

Read:

"C <SPACE> AWG-1MHz?<LF>"

→ "Control-Status[0/1]<CR+LF>"

Examples:

Read Control-Status of 1 MHz Clock "C AWG-1MHz?<LF>"

→"0<CR/LF>" (means that the external AWG 1 MHz Clock is switched OFF)

Write:

"C <SPACE> AWG-1MHz <SPACE> Control-Status[0/1]<LF>"

Examples:

Switch ON the external AWG 1 MHz Clock Reference: "C AWG-1MHz 1<LF>"

Switch OFF the external AWG 1 MHz Clock Reference: "C AWG-1MHz 0<LF>"

8.5 Standard Waveform Generation (SWG) CONTROL Commands

Several Standard Waveforms (Sine, Triangle, Sawtooth, Ramp, Pulse, Noise and DC-Voltage) can be generated directly on the LNHR DAC Ila and then transferred to one of the six Wave-Memories [WAV-A/B/E/C/D/F]. More complex user-defined waveforms can be generated easily by applying different Wave-Functions (COPY, APPEND, SUM, MULTIPLY and DIVIDE). The resulting Waveform can be saved to the volatile memory of the LNHR DAC Ila; it is called WAV-S. This saved Waveform can be recalled by selecting “Use Saved Waveform” in the SWG Mode selection.

The generated Wave-Memories can then be written to the corresponding AWG-Memories [AWG-A/B/E/C/D/F]. If needed, the corresponding Polynomial [POLY-A/B/C/D] can be applied when the Wave-Memory (reference) is written to the AWG-Memory. To do so, the Boolean parameter "Apply Polynomial" must be selected (see chapter “2D-Scan CONTROL Commands”).

8.5.1 SWG Mode (Read / Write):

Read or write the Boolean parameter SWG Mode which can be either “Generate New Waveform” (0) or “Use Saved Waveform” (1). When the default Mode “Generate New Waveform” is selected, a new waveform can be generated by using the Standard Waveforms and the Wave-Functions. In the “Use Saved Waveform” Mode the previously saved waveform (WAV-S) is recalled and the selected Wave-Functions can be applied on this recalled waveform; e.g., it can be copied to a Wave-Memory (WAV-A/B/E/C/D/F).

These two different options can be written or readout:

0=“Generate New Waveform”/1=“Use Saved Waveform”

Read:

```
"C <SPACE> SWG <SPACE> MODE?<LF>"
```

```
→ “Generate New Waveform”[0]/“Use Saved Waveform”[1]<CR+LF>"
```

Example:

Read the SWG Mode: "C SWG MODE?<LF>"

→"0<CR/LF>" (means "Generate New Waveform")

Write:

```
"C <SPACE> SWG <SPACE> MODE <SPACE> “Generate New Waveform”[0]/“Use Saved Waveform”[1]<LF>"
```

Example:

Set "Use Saved Waveform": "C SWG MODE 1<LF>"

8.5.2 Standard Waveform Function (Read / Write):

Read or write the Standard Waveform Function to be generated. The following eight different functions can be selected and they are represented by integer numbers in a range from 0 to 7:

0 = Sine function – for a Cosine function select a Phase [°] of 90°

1 = Triangle function

2 = Sawtooth function

3 = Ramp function

4 = Pulse function – the parameter Duty-Cycle [%] is applied

5 = Gaussian Noise (Fixed) – always the same seed for the random/noise-generator

6 = Gaussian Noise (Random) – random seed for the random/noise-generator

7 = DC-Voltage only – a fixed voltage is generated

Read:

"C <SPACE> SWG <SPACE> WF?<LF>"

→ "Standard Waveform[0/1/2/3/4/5/6/7]<CR+LF>"

Example:

Read the Standard Waveform: "C SWG WF?<LF>"

→"3<CR/LF>" (means Ramp function)

Write:

"C <SPACE> SWG <SPACE> WF <SPACE> Standard Waveform[0/1/2/3/4/5/6/7]<LF>"

Example:

Write a Pulse function as Standard Waveform: "C SWG WF 4<LF>"

8.5.3 Desired AWG-Frequency [Hz] (Read / Write)

Read or write the Desired AWG-Frequency [Hz] of the Wave- and AWG-function. Sometimes it isn't possible to reach exact this frequency; see also "Keep / Adapt AWG Clock-Period" and the "Nearest AWG-Frequency [Hz]". The Desired AWG-Frequency [Hz] is a floating-point number in the range between 0.001 Hz and 10'000 Hz; the decimal point must be a period (point).

At the maximum Desired AWG-Frequency of 10'000 Hz (period 100 µsec) the Standard Waveform consists of 10 points at an AWG Clock-Period of 10 µsec.

Read:

"C <SPACE> SWG <SPACE> DF?<LF>"

→ "Desired AWG-Frequency[0.001...10'000]<CR+LF>"

Example:

Read the Desired AWG Frequency: "C SWG DF?<LF>"

→"122.1<CR/LF>" (means 122.1 Hz)

Write:

"C <SPACE> SWG <SPACE> DF <SPACE> Desired AWG-Frequency[0,001...10'000]<LF>"

Example:

Write a Desired AWG-Frequency of 48.9 Hz: "C SWG DF 48.9<LF>"

8.5.4 Keep / Adapt AWG Clock-Period (Read / Write)

Read or write the Boolean parameter Keep/Adapt AWG Clock-Period. To reach the Desired AWG-Frequency as close as possible, select Adapt AWG Clock-Period.

If Adapt AWG Clock-Period (1) is selected, the AWG Clock-Period gets adapted to meet the Desired AWG-Frequency as close as possible. The update of the AWG Clock-Period on the corresponding AWG-Channel is automatically done, when the Wave-Memory is written to the AWG-Memory. See AWG Clock-Period [µsec] in the AWG CONTROL Commands.

If Keep AWG Clock-Period (0) is selected, the AWG Clock-Period of the corresponding AWG-Channel is read and used for the waveform generation. At the standard AWG Clock-Period of 10 µsec the minimal AWG Frequency is 1.538462 Hz. This is given by the maximum AWG-Memory Size of 65'000 points times the AWG Clock-Period of 10 µsec. Lower AWG frequencies can be reached by selecting higher AWG Clock-Periods.

These two different options (Keep/Adapt AWG Clock-Period) can be written or readout:
0=Keep AWG Clock-Period/1=Adapt AWG Clock-Period

Read:

```
"C <SPACE> SWG <SPACE> ACLK?<LF>"
  → "Keep/Adapt AWG-CLK[0/1]<CR+LF>"
```

Example:

Read Keep/Adapt AWG Clock-Period: "C SWG ACLK?<LF>"
→ "0<CR/LF>" (means Keep AWG Clock-Period)

Write:

```
"C <SPACE> SWG <SPACE> ACLK <SPACE> Keep/Adapt AWG-CLK[0/1]<LF>"
```

Example:

Set Adapt AWG Clock-Period: "C SWG ACLK 1<LF>"

8.5.5 Amplitude [Vp] (Read / Write)

Read or write the Amplitude [Vp] parameter of the generated Standard Waveform. This value corresponds to the peak-voltage of the generated Standard Waveform. For Gaussian-Noise the Amplitude [Vp] corresponds to the RMS-value (Sigma). The Amplitude [Vp] is a floating-point number in the range between -50.000000 V and +50.000000 V. The decimal point must be a period (point). A negative Amplitude [Vp] corresponds to a shift in Phase [°] of 180°. The ±50 V range in Amplitude [Vp] extends the flexibility in generating clipping-waveforms, also by applying a DC-Offset Voltage.

Read:

```
"C <SPACE> SWG <SPACE> AMP?<LF>"
  → "Amplitude[±50.000000]<CR+LF>"
```

Example:

Read Amplitude [Vp]: "C SWG AMP?<LF>"
→ "5.123456<CR/LF>" (means a Peak Voltage of 5.123456 V)

Write:

```
"C <SPACE> SWG <SPACE> AMP <SPACE> Amplitude[±50.000000]<LF>"
```

Example:

Write an Amplitude [Vp] of -3.253 V: "C SWG AMP -3.253<LF>"

8.5.6 DC-Offset Voltage [V] (Read / Write)

Read or write the DC-Offset Voltage [V] parameter of the generated Standard Waveform. The DC-Offset Voltage is added to the function and therefore shifts the waveform in the amplitude. If "DC-Voltage only" is selected as function, this parameter is used as fixed DC-Voltage.

The DC-Offset Voltage is a floating-point number in the range between -10.000000 V and +10.000000 V; the decimal point must be a period (point).

Read:

```
"C <SPACE> SWG <SPACE> DCV?<LF>"
  → "DC-Offset Voltage[±10.000000]<CR+LF>"
```

Example:

Read DC-Offset Voltage: "C SWG DCV?<LF>"

→ "-1.255<CR/LF>" (means a DC-Offset Voltage of -1.255 V)

Write:

"C <SPACE> SWG <SPACE> DCV <SPACE> DC-Offset Voltage[±10.000000]<LF>"

Example:

Write a DC-Offset Voltage of 3.357668 V: "C SWG DCV 3.357668<LF>"

8.5.7 Phase [°] (Read / Write)

Read or write the Phase [°] parameter of the generated Standard Waveform. The Phase shifts the generated waveform in time. It isn't applicable for Gaussian-Noise, Ramp and DC-Voltage only. A Sine with a Phase of 90° corresponds to a Cosine. The Phase [°] is a floating-point number in the range between -360.0000° and +360.0000°. The decimal point must be a period (point).

Read:

"C <SPACE> SWG <SPACE> PHA?<LF>"

→ "Phase[±360.0000]<CR+LF>"

Example:

Read the Phase: "C SWG PHA?<LF>"

→ "-120.1234<CR/LF>" (means a Phase of -120.1234°)

Write:

"C <SPACE> SWG <SPACE> PHA <SPACE> Phase[±360.0000]<LF>"

Example:

Write a Phase of +133.2568°: "C SWG PHA 133.2568<LF>"

8.5.8 Duty-Cycle [%] (Read / Write)

Read or write the Duty-Cycle [%] parameter for the generation of the Pulse-Waveform. The Duty-Cycle is only applicable for the Pulse function. A 50% Duty-Cycle results in a Square Wave; the higher the Duty-Cycle [%] the longer a high-level is applied. The Duty-Cycle [%] is a floating-point number in the range between 0.000% and 100.000%; the decimal point must be a period (point).

Read:

"C <SPACE> SWG <SPACE> DUC?<LF>"

→ "Duty-Cycle[0.000...100.000]<CR+LF>"

Example:

Read the Duty-Cycle: "C SWG DUC?<LF>"

→ "50.000<CR/LF>" (means a 50% Duty Cycle = Square Wave)

Write:

"C <SPACE> SWG <SPACE> DUC <SPACE> Duty-Cycle[0.000...100.000]<LF>"

Example:

Write a Duty-Cycle of 77.5 %: "C SWG DUC 77.5<LF>"

8.5.9 Wave-Memory Size (Read only)

Read the Wave-Memory Size of the generated Standard Waveform. This Wave-Memory Size will also be the AWG-Memory Size, after writing to the AWG-Memory. The Wave-Memory Size is an integer number in the range from 10 to 65'000 and is calculated from the Desired Frequency [Hz] parameter of the Standard Waveform Generation.

At the maximum frequency of 10'000 Hz a minimum Wave-Memory Size of 10 is reached while the AWG Clock-Period must be 10 µsec. Each point of the Wave-Memory corresponds to a DAC-Voltage in a range of ±10 V.

```
"C <SPACE> SWG <SPACE> MS?<LF>"
  → "Wave-Memory Size[10...65'000]<CR+LF>"
```

Example:

Read the Wave-Memory Size of the generated Standard Waveform: "C SWG MS?<LF>"
 → "1000<CR/LF>"

8.5.10 Nearest AWG-Frequency [Hz] (Read only)

Read the Nearest AWG-Frequency [Hz] which can be reached as close as possible to the Desired AWG-Frequency [Hz]. It is a floating-point number in the range between 0.001 Hz and 10'000 Hz.

If it must be optimized, select "Adapt AWG-CLK" (see above). Not all desired AWG-Frequencies can be achieved, since the AWG-Clock Period can only be adjusted with a resolution of 1 µsec.

```
"C <SPACE> SWG <SPACE> NF?<LF>"
  → "Nearest AWG-Frequency [0.001...10'000]<CR+LF>"
```

Example:

Read the Nearest AWG-Frequency [Hz]: "C SWG NF?<LF>"
 → "127.065<CR/LF>" (means the Nearest AWG-Frequency is 127.065 Hz)

8.5.11 Waveform Clipping Status (Read only)

Read the Waveform Clipping Status of the Generated Standard Waveform (SWG). If the amplitude of the generated waveform exceeds the maximum voltage of ± 10 V anywhere, the Clipping is set (1). If the Amplitude is always within the ± 10 V range (which means OK), the Clipping is not reset (0).

0=Not Clipping/1=Clipping

"C <SPACE> SWG <SPACE> CLP?<LF>"
 → "Waveform Clipping Status[0/1]<CR+LF>"

Example:

Read the Waveform Clipping Status: "C SWG CLP?<LF>"
 → "0<CR/LF>" (means Amplitude OK)

8.5.12 SWG/AWG Clock-Period [µsec] (Read only)

Read the SWG/AWG Clock-Period [µsec] (10...4E9), which was used for the Standard Waveform Generation (SWG). This integer number represents the AWG Clock-Period in µsec (1E-6 sec) and the resolution is 1 µsec. If "Keep AWG Clock-Period" is selected, the AWG Clock-Period of the corresponding AWG-Channel is read. If "Adapt AWG Clock-Period" is selected, the SWG/AWG Clock-Period is adapted to meet the Desired AWG Frequency as close as possible.

Read:

"C <SPACE> SWG <SPACE> CP?<LF>"
 → "AWG Clock-Period[10...4E9]<CR+LF>"

Example:

Read the AWG Clock-Period for the Standard Waveform Generation: "C SWG CP?<LF>"
 → "23<CR/LF>" (means 23 µsec)

8.5.13 Selected Wave-Memory (Read / Write)

Read or write the Selected Wave-Memory (WAV-A/B/E/C/D/F) to which the Wave-Function will be applied. If Keep AWG Clock-Period is selected above, the AWG Clock-Period of the corresponding AWG-Channel is read.

The six Wave-Memories (WAV-A/B/E/C/D/F) are represented by the following integer numbers:

- 0 = Wave-Memory A (Lower DAC-Board)
- 1 = Wave-Memory B (Lower DAC-Board)
- 2 = Wave-Memory C (Higher DAC-Board)
- 3 = Wave-Memory D (Higher DAC-Board)
- 4 = Wave-Memory E (Lower DAC-Board)
- 5 = Wave-Memory F (Higher DAC-Board)

Note: This numbering system ensures backwards compatibility with the old LNHR DAC II.

Read:

"C <SPACE> SWG <SPACE> WMEM?<LF>"
 → "Selected Wave-Memory[0/1/2/3/4/5]<CR+LF>"

Example:

Read the Selected Wave-Memory: "C SWG WMEM?<LF>"
 → "4<CR/LF>" (means Wave-Memory E is selected)

Write:

"C <SPACE> SWG <SPACE> WMEM <SPACE> Selected Wave-Memory[0/1/2/3/4/5]
 <LF>"

Example:

Select Wave-Memory D for applying the Wave-Function: "C SWG WMEM 3<LF>"

8.5.14 Selected Wave-Function (Read / Write)

Read or write the Selected Wave-Function which will be applied on the generated Standard Waveform and the Selected Wave-Memory when "Apply to Wave-Memory Now" is operated.

The following Wave-Functions are available: COPY, APPEND, SUM, MULTIPLY and DIVIDE. When COPY Waveform is selected, the actual Wave-Memory is overwritten. The other four Wave-Functions can be applied to START or to the END of Wave-Memory.

With these Wave-Functions complex and user-specific waveforms can be created in the Wave-Memory. Multiple Wave-Functions can be applied on different generated Standard Waveforms to reach the desired user-specific waveform.

These nine different Wave-Functions are available and are represented by the following integer numbers:

- 0 = COPY to Wave-MEM -> Overwrite
- 1 = APPEND to Wave-MEM @START
- 2 = APPEND to Wave-MEM @END
- 3 = SUM Wave-MEM @START
- 4 = SUM Wave-MEM @END
- 5 = MULTIPLY Wave-MEM @START
- 6 = MULTIPLY Wave-MEM @END
- 7 = DIVIDE Wave-MEM @START
- 8 = DIVIDE Wave-MEM @END

Read:

"C <SPACE> SWG <SPACE> WFUN?<LF>"
 → "Selected Wave-Function[0...8]<CR+LF>"

Example:

Read the Selected Wave-Function: "C SWG WFUN?<LF>"
 → "4<CR/LF>" (means the Wave-Function "SUM Wave-MEM @END" is selected)

Write:

"C <SPACE> SWG <SPACE> WFUN <SPACE> Selected Wave-Function[0...8]<LF>"

Examples:

Select the Wave-Function "COPY to Wave-MEM -> Overwrite": "C SWG WFUN 0<LF>"
 Select the Wave-Function "MULTIPLY Wave-MEM @START": "C SWG WFUN 5<LF>"

8.5.15 No Linearization / Linearization for actual DAC-Channel (Read / Write)

Read or write the Boolean parameter No Linearization/Linearization for actual DAC-Channel. When “Copy to Wave-Memory Now” is performed, the actual selected AWG DAC-Channel gets registered for the later linearization when the WAV-Memory is written to the AWG-Memory. In this case, the AWG-Waveform gets also linearized after applying the polynomial. If No Linearization is selected the DAC-Channel is registered as a zero (0) which indicates that no linearization will be done when the WAV-Memory is written to the AWG-Memory. This slightly increases the write performance.

These two options (No Linearization/Linearization for actual DAC-Channel) can be written or readout:

0=No Linearization/1=Linearization for actual DAC-Channel

Read:

"C <SPACE> SWG <SPACE> LIN?<LF>"

→ “No Linearization/Linearization for actual DAC-Channel[0/1]<CR+LF>”

Example:

Read No Linearization/Linearization for actual DAC-Channel: “C SWG LIN?<LF>”

→”0<CR/LF>” (means No Linearization)

Write:

"C <SPACE> SWG <SPACE> LIN <SPACE>

No Linearization/Linearization for actual DAC-Channel[0/1]<LF>"

Example:

Set Linearization for actual DAC-Channel: “C SWG LIN 1<LF>”

8.5.16 Apply Wave-Function to Wave-Memory Now (Write only)

The Selected Wave-Function gets applied to the Selected Wave-Memory (WAV-A/B/E/C/D/F). At this moment, the actual selected AWG DAC-Channel gets registered if “Linearization for actual DAC-Channel” is selected (see above). After setting this control, it gets reset internally.

"C <SPACE> SWG APPLY<LF>"

Example:

Apply the Selected Wave-Function to the Selected Wave-Memory: “C SWG APPLY<LF>”

8.6 Wave CONTROL Commands

Some parameters of the six [seven] Wave-Memories (WAV-A/B/E/C/D/F/[S]) can be readout by these Wave CONTROL Commands. By using the WRITE command, the Wave-Memory (reference) can be written to the corresponding AWG-Memory (AWG-A/B/E/C/D/F). With the SAVE command, the content of one of the Wave-Memories (WAV-A/B/E/C/D/F) can be saved to the internal volatile memory on the LNHR DAC IIa; it is called WAV-S. The six Wave-Memories (WAV-A/B/E/C/D/F) as well as the Saved Wave-Memory (WAV-S) can be cleared by using the Wave-Memory Clear (CLR) function.

8.6.1 Wave-Memory Size (Read only)

Read the Size of one of the seven Wave-Memories (WAV-A/B/E/C/D/F/S). This Wave-Memory Size will also be the AWG-Memory Size, after writing to the AWG-Memory. The Wave-Memory Size is an integer number in the range from 0 to 65'000. A Wave-Memory Size of zero (0) indicates that this Wave-Memory is cleared.

Note: For optimal performance, keep the Wave-Memory Size as small as possible and always clear unused Wave-Memories.

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> MS?<LF>"
  → "Wave-Memory Size[0...65'000]<CR+LF>"
```

Example:

Read the Size of Wave-Memory B: "C WAV-B MS?<LF>"
→"1000<CR/LF>"

8.6.2 Wave-Memory Clear (Write only)

Clear the selected Wave-Memory (WAV-A/B/E/C/D/F/S) and set the Wave-Memory Size to zero (0). The WAV-S is the Saved Waveform. After setting this control, it gets reset internally.

Note: For optimal performance, always clear unused Wave-Memories.

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F/S] <SPACE> CLR<LF>"
```

Example:

Clear the Wave-Memory B: "C WAV-B CLR<LF>"

8.6.3 Wave-Memory Save (Write only)

Save the selected Wave-Memory (WAV-A/B/E/C/D/F) to the internal volatile memory on the LNHR DAC IIa; it is called WAV-S.

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F] <SPACE> SAVE<LF>"
```

Example:

Save the Wave-Memory F to internal WAV-S: "C WAV-F SAVE<LF>"

8.6.4 Wave-Memory Linearization DAC-Channel (Read only)

Read the corresponding DAC-Channel for the Linearization of one of the six Wave-Memories (WAV-A/B/E/C/D/F). The Linearization for this DAC-Channel is done when the Wave-Memory is written to the AWG-Memory. If "No Linearization" was selected

when the waveform was copied to the Wave-Memory, the corresponding DAC-Channel is 0 (zero). The DAC-Channel for Linearization is an integer number in the range from 0 to 24. The not existing DAC-Channel 0 (zero) means that no linearization gets applied.

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F] <SPACE> LINCH?<LF>"
  → "DAC-Channel[0...24]<CR+LF>"
```

Example:

Read the DAC-Channel for Linearization of Wave-Memory D: "C WAV-D LINCH?<LF>"
→"23<CR/LF>"

8.6.5 Write Wave-Memory to AWG-Memory (Write only)

Write the Wave-Memory (WAV-A/B/E/C/D/F) to the corresponding AWG-Memory (AWG-A/B/E/C/D/F). After setting this control, it gets reset internally. The Polynomial is only applied when the corresponding Boolean parameter "Apply Polynomial" is selected (see chapter "2D-Scan CONTROL Commands").

Note: The Saved Waveform (WAV-S) has first to be copied to one of the six Wave-Memories (WAV-A/B/E/C/D/F) before it can be written to the corresponding AWG-Memory. To recalled the Saved Waveform (WAV-S) select "Use Saved Waveform" in the SWG Mode selection.

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F] <SPACE> WRITE<LF>"
```

Example:

Write the Wave-Memory E to the AWG-Memory C: "C WAV-E WRITE<LF>"

8.6.6 Wave-Memory Writing Busy (Read only)

During writing the Wave-Memory (WAV-A/B/E/C/D/F) to the corresponding AWG-Memory (AWG-A/B/E/C/D/F) a Busy flag is set (1). If it is Idle state the value is zero (0). 0=Idle/1=Busy (writing WAV- to AWG-Memory)

```
"C <SPACE> WAV-Memory[WAV-A/B/E/C/D/F] <SPACE> BUSY?<LF>"
  → "Idle/Busy[0/1]<CR+LF>"
```

Example:

Read Writing Busy of Wave-Memory E: "C WAV-E BUSY?<LF>"
→"0<CR/LF>" (means Idle = not writing to AWG-Memory C)

CAUTION: It takes around 200 msec for the Busy flag to be set. Therefore, after issuing a WRITE command, wait at least 200 msec before polling the Busy flag.

9 Converting DAC-Voltage to DAC-Value

A DAC-Value is a 24-bit number in the decimal range from 0 to 16'777'215 ($2^{24}-1$); this corresponds to a hexadecimal range from 0x000000 to 0xFFFFFFFF.

The DAC-Voltage has a fixed range from -10 V to +10 V with a step-size of 1.192093 μ V ($20 \text{ V} / 16'777'215$).

For a given DAC output voltage (V_{out} [-10 V ...+10 V]) the 24-bit decimal DAC-Value (DACval [0...16'777'215]=[0x000000...0xFFFFFFFF]) is given by (rounded to the next integer value):

$$\text{DACval}_{\text{dec}} = (V_{\text{out}} + 10) \cdot 838'860.74$$

To get a DAC-Value (HEX), which is needed for remote programming the DAC output voltage, the decimal number has to be converted to a hexadecimal number. All higher program languages have already included such a conversion-function.

For a given decimal DAC-Value ($\text{DACval}_{\text{dec}}$ [0...16'777'215]=[0x000000...0xFFFFFFFF]) the DAC output voltage (V_{out} [-10 V...+10 V]) can be determined by:

$$V_{\text{out}} = (\text{DACval}_{\text{dec}} / 838'860.74) - 10$$

The table below shows the DAC-Voltage [$\pm 10 \text{ V}$] in 1 V steps and the calculated DAC-Value (decimal) and the corresponding DAC-Value (HEX):

DAC-Voltage	DAC-Value (decimal)	DAC-Value (HEX)
+10 V	16'777'215	0xFFFFFFFF
+9 V	15'938'354	0xF33332
+8 V	15'099'493	0xE66665
+7 V	14'260'633	0xD99999
+6 V	13'421'772	0xCCCCC
+5 V	12'582'911	0xBFFFFFF
+4 V	11'744'050	0xB33332
+3 V	10'905'190	0xA66666
+2 V	10'066'329	0x999999
+1 V	9'227'468	0x8CCCCC
0 V	8'388'607	0x7FFFFFF
-1 V	7'549'747	0x733333
-2 V	6'710'886	0x666666
-3 V	5'872'025	0x599999
-4 V	5'033'164	0x4CCCCC
-5 V	4'194'304	0x400000
-6 V	3'355'443	0x333333
-7 V	2'516'582	0x266666
-8 V	1'677'721	0x199999
-9 V	838'861	0x0CCCCD
-10 V	0	0x000000

10 Compilation of all Commands with “Examples”

5 SET COMMANDS (WRITE)

5.1 SET DAC Commands

5.1.1/2 SET DAC-Channel/ALL to a registered DAC-Value (HEX)	DAC-CH#/ALL Value(HEX) “1 7FFFFF”
5.1.3/4 SET DAC-Channel/ALL Status (ON/OFF)	DAC-CH#/ALL ON/OFF “4 ON”
5.1.5/6 SET DAC-Channel/ALL Bandwidth (LBW/HBW)	DAC-CH#/ALL LBW/HBW “6 HBW”

5.2 SET AWG Commands

5.2.1/2 SET an AWG-Memory Adr(HEX)/ALL to a Value (HEX)	AWG-A...F Adr(HEX)/ALL Value(HEX) “AWG-B 0025 8CCCCC”
---	--

5.3 SET WAVE Commands

5.3.1/2 SET a WAV-Memory Adr(HEX)/ALL to a Voltage (V)	WAV-A...F/S Adr(HEX)/ALL Volt(Float) “WAV-C 84CF -8.881717”
--	--

5.4 SET POLYNOMIAL Commands

5.4.1 SET the POLYNOMIAL Coefficients	POLY-A...D POLY-Coefficients(Float) “POLY-B 0.1 2.1 3E-3”
---------------------------------------	--

6 MULTIPLE SET COMMANDS (WRITE)

Multiple (up to 1'000) SET-Commands separated by a semicolon (“;”) Set-Command 1;Set-Command 2;...	“1 8CCCCC;2 A99999;3 F66666;1 ON”
--	-----------------------------------

7 QUERY COMMANDS (READ)

7.1 QUERY DATA Commands

7.1.1/2 QUERY an Actual DAC-Value (HEX) from DAC-Channel/ALL	CH#/ALL V? “2 V?”
7.1.3/4 QUERY a Registered DAC-Value (HEX) from DAC-Channel/ALL	CH#/ALL VR? “2 VR?”
7.1.5/6 QUERY a DAC-Status (ON/OFF) from DAC-Channel/ALL	CH#/ALL S? “12 S?”
7.1.7/8 QUERY a DAC-Bandwidth (LBW/HBW) from DAC-Channel/ALL	CH#/ALL BW? “6 BW?”
7.1.9/10 QUERY a DAC-MODE (ERR/DAC/ ...) from DAC-Channel/ALL	CH#/ALL M? “17 M?”
7.1.11 QUERY an AWG-Value (HEX) at AWG-Address (HEX)	AWG-A...F Address(HEX)? “AWG-C 84CF?”
7.1.12 QUERY a 1000-Block of AWG-Value (HEX) at StartAdr (HEX)	AWG-A...F StartAdr(HEX) BLK? “AWG-B 3ABC BLK?”
7.1.13 QUERY a WAV-Value (HEX) at Address (HEX)	WAV-A...F/S Address(HEX)? “WAV-B 12A3?”
7.1.14 QUERY a 1000-Block of WAV-Voltage (V) at StartAdr (HEX)	WAV-A...F/S StartAdr(HEX) BLK? “WAV-D 12AB BLK?”
7.1.15 QUERY the POLYNOMIAL Coefficients	POLY-A...D? “POLY-B?”

7.2 QUERY INFORMATION Commands

7.2.1 QUERY overview of the ASCII Commands	“?”
7.2.2 QUERY HELP	“HELP?”
7.2.3 QUERY the Software Release on the LNHR DAC IIa	“SOFT?”
7.2.4 QUERY the Hardware information	“HARD?”
7.2.4 QUERY the device identification with serial number	“IDN?”
7.2.6 QUERY the Health information	“HEALTH?”
7.2.7 QUERY the actual IP-Address and Subnet-Mask	“IP?”
7.2.8 QUERY the actual Baud-Rate of the RS-232 port	“SERIAL?”
7.2.9 QUERY the contact address in case of problems	“CONTACT?”

8 CONTROL COMMANDS (READ/WRITE)

8.1 DAC Update-Mode and Synchronization CONTROL Commands

8.1.1 DAC Update-Mode: 0=Normal/1=SYNC (Read/Write)

C UM-L/H(?) 0/1
Read: "C UM-L?"
Write: "C UM-L 1"

8.1.2 Synchronous DAC-Update-Mode (Write only)

C SYNC-L/H/LH
"C SYNC-LH"

8.2 RAMP/STEP-Generator CONTROL Commands

8.2.1 RAMP Start/Hold/Stop (Write only)

C RMP-A...D/ALL START/HOLD/STOP
"C RMP-A START"

8.2.2 RAMP State: 0...3 => Idle/R_UP/R_DOWN/Hold (Read only)

C RMP-A...D S?
"C RMP-B S?"

8.2.3 RAMP Cycles-Done: 0...4E9 (Read only)

C RMP-A...D CD?
"C RMP-D CD?"

8.2.4 RAMP Steps-Done: 0...4E9 (Read only)

C RMP-A...D SD?
"C RMP-A SD?"

8.2.5 RAMP Step-Size Voltage: ± 10 V (Read only)

C RMP-A...D SSV?
"C RMP-C SSV?"

8.2.6 RAMP Steps per Cycle: 10...2E8 (Read only)

C RMP-A...D ST?
"C RMP-A ST?"

8.2.7 RAMP DAC-Channel AVAILABLE: 0=Not Ava/1=Ava (Read only)

C RMP-A...D AVA?
"C RMP-B AVA?"

8.2.8 RAMP Selected DAC-CHANNEL: 1...24 (Read/Write)

C RMP-A...D CH(?) DAC-CH#
Read: "C RMP-A CH?"

8.2.9 RAMP Start Voltage: ± 10 V (Read/Write)

Write: "C RMP-A CH 10"
C RMP-A...D STAV(?) Volt(Float)

8.2.10 RAMP Stop/Peak Voltage: ± 10 V (Read/Write)

Read: "C RMP-B STAV?"
Write: "C RMP-B STAV -2.135"

8.2.11 RAMP Time: 0.05...1E6 sec (Read/Write)

C RMP-A...D STOV(?) Volt(Float)

8.2.12 RAMP Shape: 0=UP-ONLY/1=UP-DOWN (Read/Write)

Read: "C RMP-C STOV?"
Write: "C RMP-C STOV 5.128"

8.2.13 RAMP Cycles-Set: 0 (Inf)...4E9 (Read/Write)

C RMP-A...D RT(?) Time(sec,Float)

8.2.14 RAMP/STEP- Selection: 0=RAMP/1=STEP (Read/Write)

Read: "C RMP-A RT?"
Write: "C RMP-A RT 10.155"

8.3 2D-Scan CONTROL Commands

8.3.1 Normal-/Auto-Start: 0=Normal/1=Auto-Start (Read/Write)

C RMP-A...D RS(?) 0/1
Read: "C RMP-C RS?"
Write: "C RMP-C RS 1"

8.3.2 Keep-/Reload-AWG MEM: 0=Keep/1=Reload (Read/Write)

C RMP-A...D CS(?) Number(DEC)

8.3.3 Skip-/Apply-Polynomial: 0=Skip/1=Apply (Read/Write)

Read: "C RMP-D CS?"
Write: "C RMP-D CS 112"

8.3.4 Adaptive Shift-Voltage per Step: ± 10 V (Read/Write)

C RMP-A...D STEP(?) 0/1
Read: "C RMP-A STEP?"
Write: "C RMP-A STEP 1"

C AWG-A...D AS(?) 0/1
Read: "C AWG-C AS?"
Write: "C AWG-C AS 1"

C AWG-A...D RLD(?) 0/1
Read: "C AWG-B RLD?"
Write: "C AWG-B RLD 0"

C AWG-A...D AP(?) 0/1
Read: "C AWG-A AP?"
Write: "C AWG-A AP 1"

C AWG-A...D SHIV(?) Volt(Float)

Read: "C AWG-B SHIV?"
Write: "C AWG-B SHIV 1E-3"

8.4 AWG CONTROL Commands

8.4.1 AWG Normal/AWG Only: 0=Normal/1=AWG Only (Read/Write)	C AWG-AB(E)/CD(F) ONLY(?) 0/1 Read: "C AWG-AB ONLY?" Write: "C AWG-CD ONLY 1"
8.4.2 AWG Start/Stop (Write only)	C AWG-A...F/AB/ABE/CD/CDF/ALL Start/Stop "C AWG-ALL STOP"
8.4.3 AWG State: 0=Idle/1=Running (Read only)	C AWG-A...F S? "C AWG-B S?"
8.4.4 AWG Cycles-Done: 0...4E9 (Read only)	C AWG-A...F CD? "C AWG-F CD?"
8.4.5 AWG Duration/Period: 20E-6...2.6E8 sec (Read only)	C AWG-A...F DP? "C AWG-E DP?"
8.4.6 AWG DAC-Channel AVAILABLE: 0=Not Ava/1=Ava (Read only)	C AWG-A...F AVA? "C AWG-D AVA?"
8.4.7 AWG Selected DAC-CHANNEL: L=1...12/H=13...24 (Read/Write)	C AWG-A...F CH(?) DAC-CH# Read: "C AWG-B CH?" Write: "C AWG-D CH 24"
8.4.8 AWG-Memory Size: 2...65'000 (Read/Write)	C AWG-A...F MS(?) Number(DEC) Read: "C AWG-A MS?" Write: "C AWG-B MS 1000"
8.4.9 AWG Cycles-Set: 0 (Inf)...4E9 (Read/Write)	C AWG-A...F CS(?) Number(DEC) Read: "C AWG-D CS?" Write: "C AWG-B CS 23"
8.4.10 AWG External Trigger Mode: 0/1/2/3 (Read/Write)	C AWG-A...F TM(?) 0/1/2/3 Read: "C AWG-A TM?" Write: "C AWG-E TM 0"
8.4.11 AWG Clock-Period [μ sec]: 10...4E9 μ sec (Read/Write)	C AWG-A...F, AB/CD/ABE/CDF CP(?) Period(μ sec,DEC) Read: "C AWG-AB CP?" Write: "C AWG-F CP 200"
8.4.12 AWG 1 MHz CLK Ref ON/OFF: 0=OFF/1=ON (Read/Write)	C AWG-1MHz(?) 0/1 Read: "C AWG-1MHz?" Write: "C AWG-1MHz 0"

8.5 Standard Waveform Generation CONTROL Commands

8.5.1 Generate New WF (0)/Use Saved WF (1): 0/1 (Read/Write)	C SWG MODE(?) 0/1 Read: "C SWG MODE?" Write: "C SWG MODE 1"
8.5.2 Standard Waveform Function: 0...7 (Read/Write)	C SWG WF(?) 0...7 Read: "C SWG WF?" Write: "C SWG WF 2"
8.5.3 Desired AWG Frequency [Hz]: 0.001...10'000 (Read/Write)	C SWG DF(?) Frequency(Hz,Float) Read: "C SWG DF?" Write: "C SWG DF 73.45"
8.5.4 Keep/Adapt AWG Clock-Period: 0=Keep/1=Adapt (Read/Write)	C SWG ACLK(?) 0/1 Read: "C SWG ACLK?" Write: "C SWG ACLK 0"
8.5.5 Amplitude [Vp]: ± 50 Vp (Read/Write)	C SWG AMP(?) Amplitude(Vp,Float) Read: "C SWG AMP?" Write: "C SWG AMP 0.277"
8.5.6 DC-Offset Voltage [V]: ± 10 V (Read/Write)	C SWG DCV(?) DC-Offset(Voff,Float) Read: "C SWG DCV?" Write: "C SWG DCV -1.88"
8.5.7 Phase [$^{\circ}$]: $\pm 360^{\circ}$ (Read/Write)	C SWG PHA(?) Phase($^{\circ}$,Float) Read: "C SWG PHA?" Write: "C SWG PHA 75.8"
8.5.8 Duty-Cycle [%]: 0...100 (Read/Write)	C SWG DUC(?) Duty-Cycle(%,Float) Read: "C SWG DUC?" Write: "C SWG DUC 32.155"

8.5.9 Generated SWG-Memory Size: 10...65'000 (Read only)	C SWG MS? "C SWG MS?"
8.5.10 Nearest AWG-Frequency [Hz]: 0.001...10'000 (Read only)	C SWG NF? "C SWG NF?"
8.5.11 Waveform Clipping Status: 0=Not Clip/1=Clip (Read only)	C SWG CLP? "C SWG CLP?"
8.5.12 SWG/AWG Clock-Period [μ sec]: 10...4E9 μ sec (Read only)	C SWG CP? "C SWG CP?"
8.5.13 Selected Wave-Memory: 0/1/2/3/4/5 = A/B/C/D/E/F (R/W)	C SWG WMEM(?) 0/1/2/3/4/5 Read: "C SWG WMEM?" Write: "C SWG WMEM 4"
8.5.14 Selected Wave-Function: 0...8 (Read/Write)	C SWG WFUN(?) 0...8 Read: "C SWG WFUN?" Write: "C SWG WFUN 4"
8.5.15 No/Linearization for actual DAC-Channel: 0/1 (Read/Write)	C SWG LIN(?) 0/1 Read: "C SWG LIN?" Write: "C SWG LIN 1"
8.5.16 Apply Wave-Function to Wave-Memory Now (Write only)	C SWG APPLY "C SWG APPLY"
8.6 Wave CONTROL Commands	
8.6.1 Wave-Memory Size: 0...65'000 (Read only)	C WAV-A...F/S MS? "C WAV-A MS?"
8.6.2 Wave-Memory Clear (Write only)	C WAV-A...F/S CLR "C WAV-B CLR"
8.6.3 Wave-Memory Save (Write only)	C WAV-A...F SAVE "C WAV-C SAVE"
8.6.4 Wave-Memory Linearization DAC-Channel: 0...24 (Read only)	C WAV-A...F LINCH? "C WAV-D LINCH?"
8.6.5 Write Wave-Memory to AWG-Memory (Write only)	C WAV-A...F WRITE "C WAV-E WRITE"
8.6.6 Wave-Memory Writing Busy: 0=Idle/1=Busy (Read only)	C WAV-A...F BUSY? "C WAV-F BUSY?"