Controller for Programmable High-Voltage Power Supplies

Firmware Version 1-10

User Manual

Document version A, created on Dec-09-2022

CGC Instruments
Hübschmannstr. 18 | D–09112 Chemnitz
Tel.: +49 (371) 355 098–55
Fax: +49 (371) 355 098–60
internet: www.cgc-instruments.com
e–mail: info@cgc-instruments.com

Contents

Figure List

Table List

Safety Information

- The device may be installed and used by authorized and instructed personnel only. Read this manual carefully before installing and using the device. Always follow the safety notes and warnings in this manual.
- The device is designed for indoor dry laboratory use only. Before powering the device on, the device temperature must accommodate to the ambient temperature to avoid moisture condensation. This is especially relevant after transportation.
- Do not operate the device if it is damaged or not functioning properly. Never use damaged cables or accessories.
- Do not open the device case, install replacement parts, or perform modifications to the device. There are no user serviceable parts inside.
- To avoid damage, connect the line cord to a properly wired and grounded receptacle only. Be sure that the mains voltage and the fuse rating match the device specification. Never operate the device during thunderstorms.
- Never use corrosive or abrasive cleaning agents or polishes, avoid the usage of organic solvents. If necessary, clean the device with a soft moist cloth. Make sure that the device is completely dry and free from contaminants before powering it on.

! **Warning:** The voltage controller is an electronic device that is sensitive to electrostatic electricity. While manipulating with the controller, the ESD (*Electro-Static Discharge*) protection rules must be kept in mind.

Technical Data

Characteristics

- control of four high-voltage amplifiers and a dual high-voltage switching converter
- integrated power supply unit
- supply voltage supervising
- output voltage and temperature monitoring, fan controller
- controller of an inrush-current limiter
- serving of two independent interlock loops, selectable per software
- 16-bit RISC microcontroller running at 16 MHz
- non-volatile memory, data space: 1 MB
- user-friendly graphical interface
- USB data interface

Power Supply Unit

- technology: switching mode with output LC filters and input rectifier
- current limiter, over-voltage and over-temperature protection
- output voltages:
  12 V, 5 A max., voltage tolerance ±2%, efficiency >90%
  5 V, 3 A max., voltage tolerance ±1%, efficiency >90%
  3.3 V, 3 A max., voltage tolerance ±1%, efficiency >95%

Human-Machine Interface (HMI)

- monochrome LCD display 128x64 pixel
  pixel size: 0.5 mm
  pixel color: yellow, background: blue
  background illumination: white LED
- keypad: 5 keys: 4x direction + 1x "enter"
- rotary encoder: 24 positions per revolution, integrated press button
- optional external shutdown button via the interlock loop

Digital Interface

- USB interface according to USB 2.0 standard
  connector: USB plug type B
  data transfer rate: up to 12 MBit/s (*Full Speed*)
  effective data transfer rate: >100 kBit/s

Description

General

The voltage controller is the control module of programmable high-voltage power supplies integrated in a 19" case. It provides control signals and a part of supply voltages for four high-voltage amplifiers and two high-voltage switching converters. The controller is equipped with a USB data interface. It allows to transfer data to or from the device and to control it remotely (see sections "Software Utilities" and "Software Interface"). The controller also integrates a switching-mode power supply unit that provides supply voltages of 12 V, 5 V, and 3.3 V for digital circuits of the device.

The device output voltages are generated in several modules - high-voltage amplifiers and high-voltage switching converters. The voltage controller controls these modules and supervises their operation parameters. The outputs are continuously monitored and, in case of any discrepancy, the device is deactivated. This can happen if the outputs are overloaded, the shutdown protects the attaches hardware from secondary damages.

The controller monitors two interlock loops, it disables the outputs of the high-voltage amplifiers and the high-voltage switching converters if any of the enabled loops is opened. The terminals of the interlock loops are located at two BNC sockets at device front and rear panels, respectively. They can be connected to a vacuum controller or other supervising device that opens the contacts in emergency states. The function of each particular interlock loop can be disabled if it is not used. If both interlock loops are disabled, the device can be activated regardless of the interlock state.

The voltage controller monitors several temperature sensors, which measure the temperature of the internal heatsinks and some integrated circuits. The measured temperatures are used to control a fan at the rear panel that actively cools the device. The fan function is monitored and a warning light (red LED "Failure") is activated, if the fan does not operate at the desired speed. On such failures, the operation of the device is still possible but the temperature may rise and the performance degrade. The controller disables the device only if it overheats or if any temperature sensor fails.

The actual settings of the output voltages and other operation parameters are stored in a non-volatile memory and are automatically restored at the next device startup.

The voltage controller contains a microcontroller that serves a human-machine interface (see section "Human-Machine Interface"). The microcontroller is connected to an FPGA that implements the hardware interfaces to other modules of the device. The microcontroller's firmware realizes a user-friendly graphical interface. The device is controlled by the help of menus and dialog boxes, they provide an intuitive device control (see section "User Interface").

Control and Indication Elements

All control elements of the device are located on the front panel (see Fig. 1).

The rocker switch "Power" turns the device on or off. If the device is powered on, the LCD is activated and its backlight is turned on. Manufacturer logo is shown and, after a short hardware initialization, the device enters its idle state where it waits for user commands (see Fig. 2).

If the device is activated, the high-voltage supply, the amplifiers, and the high-voltage switching converters are turned on. This state is signalized by the green LED " Activated".

The yellow LED "Interlock" shows the state of the interlock loops. It lights if the enabled interlock loops are closed. When any of the enabled loops is opened, the LED "Interlock" is off and the device cannot be activated. If it was active, an opened interlock loop shuts the hardware down.

The red LED "Failure" lights if any failure occurs. This may be a device overheating or a too low device temperature[†], a failure of the supply voltages or of the fan. Only the fan failure does not prevent the device from operation, on all other failures, the device cannot be activated. The reason of the failure can be found out by inspecting the dialog boxes for device monitoring (see section "Device Monitoring"). If a fan failed, the device continues to operate, you should, however, replace the defective fan as soon as possible to prevent overheating.

[†] The device cannot be activated at temperatures below 10°C because of a possible moisture condensation.

The temperature thresholds for the fan operation and for overheating can be adjusted (see section "Device Setup"). If an overheating event occurs, check these thresholds and adjust them if necessary.

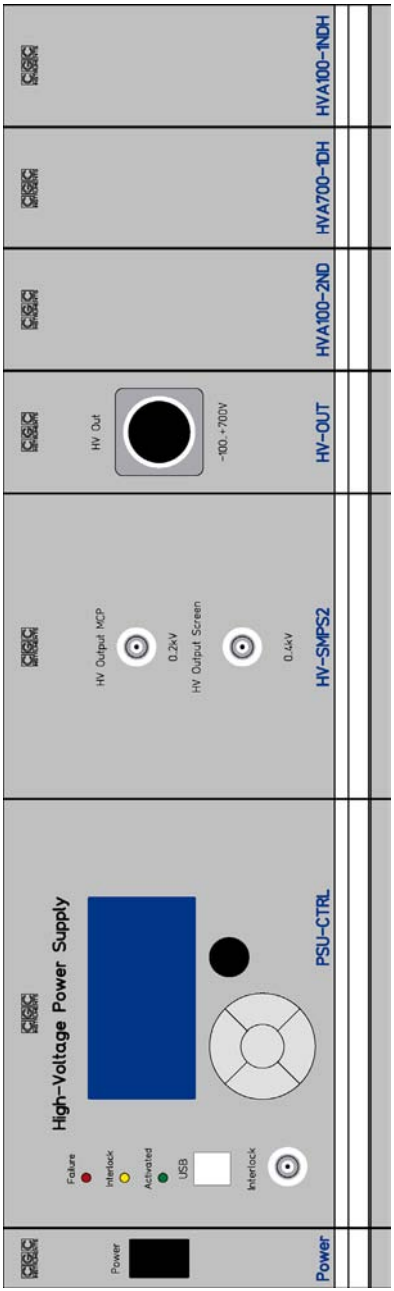

Fig. 1. Front panel of the programmable power supply unit.

Quick Setup Guide

To prepare the device for an experiment, you need just to connect all cables to the experimental setup. You should, however, check the operation parameters before connecting it. To do so, follow the next steps:

- Ensure that the proper line voltage is selected (consult the manual of the main device), connect the line cord and turn the device on. The LCD should light up.
- Press the middle key "enter" and start the system control by selecting the menu items under "Setup ► System" (see section "Fan Preferences"). Check the displayed values and, if desired, change them. Finally, close the dialog box.
- Press the middle key "enter" and start the interlock setup by selecting the menu items under "Setup ► Interlock". Check the displayed values and if desired change them (for details, see section "Interlock Setup"). Close the dialog box and close the selected interlock connector, for instance, by a BNC termination plug. Alternatively, disable both interlock loops, in such case, the interlock connectors may remain open. The LED "Interlock" should light up. If it does not happen, check the interlock state by pressing the middle key "enter" and selecting the menu item "Monitor ► Interlock" (see section "Interlock Monitor").
- Press the middle key "enter" and start the supply monitoring by selecting the menu item "Monitor ► Supply" (see section "Supply Monitor"). Check the displayed values and close the dialog box if the values do not differ from the expected values significantly.
- Press the middle key "enter" and start the voltage control by selecting the menu item "Setup ► Voltage" (see section "Voltage Control"). Check that al voltages are set to 0. If necessary, select the particular voltage value by the vertical keys and reset it by pressing the encoder knob. Select the item "HV output" and change its state to "enabled" to activate the device. The LED "Activated" should light up. If it does not happen, check the interlock state and other device parameters in the submenu "Monitor" (see section "Device Monitoring"). It the device could be activated, set the desired output voltages and check their real values. The dialog box "Voltage Control" shows the measured value of the output voltage above the particular control value.

- Select the item "HV output" and change its state to "disabled" to deactivate the device. The LED "Activated" should be shut off. Finally, close the dialog box.
- Connect the device to a PC via a USB cable[‡]. If necessary, install the device driver (see section "Driver Installation"). Check the port number to which the device is attached in the PC. Locate the utility **`RemoteControl.exe`** in the software package and launch it with the port number as a command-line parameter (for more details, see section "Utility RemoteControl"). Using the utility, you should be able to control the device remotely by pressing the arrow keys and enter.

If you encounter problems, read carefully the corresponding section in this manual. If you cannot solve the problem, contact the manufacturer of the device. Before doing it, obtain hardcopies of the LCD panel (see section "Utility RemoteControl") showing all monitoring parameters (see section "Device Monitoring") and send them together with the problem description.

[‡] The USB cable is not a part of the device and has to be ordered separately. For the connection to a PC you may use any standard USB cable with the connectors A and B and a length of 5 m maximum.

User Interface

Human-Machine Interface

The device is equipped with a graphic liquid crystal display (LCD), a set of keys (keypad) and a rotary encoder. The keys are arranged in a circle symbolizing the key function: There are four direction keys ("left", "right", "up", and "down") and a middle key for confirmation or selection ("enter").

The keys are used for menu navigation, for selecting dialog items, or for changing values. The function of the keypad in every device state is symbolized on the LCD immediately above the keypad. A text describes the function of the middle key, alternatively just the symbol "↵" is displayed showing that the middle key is functional. Similarly, arrows show which of the keys has an influence on the operation in the current state. When a menu is active, the vertical direction keys are used to change the current selection. The right direction key as well the middle key opens a submenu, provided it is available. The left direction key closes the submenu or the main menu if there was no opened submenu. The middle key selects the menu item and launches the corresponding action.

The rotary encoder is used to change numerical values or select items. The function of the encoder is symbolized on the LCD immediately above the encoder knob in every device state. In most situations, the encoder uses an enhanced speed control. This enables you to precisely set any desired value or quickly make large changes. You can change the corresponding value in small steps when you rotate the encoder slowly or make large changes when you spin the encoder knob rapidly. To set large numbers, the encoder speed can be further increased by pressing and holding the direction key "left". If this feature is available, the left arrow is shown above the keys.

In some situations, the encoder's push button is used to reset the selected value, set it to default, or switch the encoder function. In such cases, the symbol "⧈" is shown on the LCD above the encoder knob reporting that the encoder's push button is functional.

Main Program Menu

When the device starts, the initialization takes place and the manufacturer logo is shown for couple of seconds. Then, the device enters the idle state (see Fig. 2). The lower right corner shows the uptime, i.e. the time elapsed from the last device start.

Fig. 2. Program display in device idle state.

By pressing the middle key "enter", the main menu can be accessed (see Fig. 3). You can control the device (submenu "Control", see also section "Device Control"), setup the device settings (submenu "Setup", see also section "Device Setup"), monitor the device function (submenu "Monitor", see also section "Device Monitoring"), view the device state (submenu "View", see also section "Device State"), or manage the system settings (submenu "Adjustment").

You can save the current system settings (menu item "Adjustment ► Save"), load the settings back (menu item "Adjustment ► Load"), or reset them to default values (menu item "Adjustment ► Reset"). The system settings are saved automatically to a non-volatile memory when the device powers down and are loaded back when the device starts.

You may save the settings using the main menu prior to the next device shutdown if you wish to modify the settings, but have the possibility to restore the current state by loading the saved values back. When you choose to load the settings back, the settings will be reloaded and the last saved state will be restored, i.e. the settings will be set to the state immediately before the last shutdown or after the last startup, or to the state when you saved the settings. Further, you may reset the settings to their default values. This is advisable if you wish to restore the original device state.

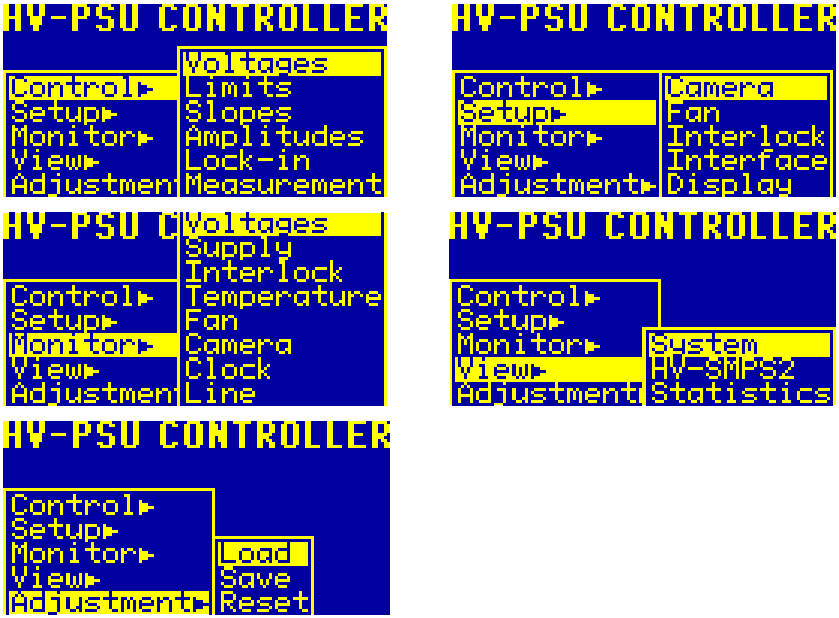

Fig. 3. Main program menu.
The figure shows the different submenus of the main program menu.

Device Control

Several dialog boxes are available to control the device.

Voltage Control

Using the dialog box "Voltage Control", you can set the output voltages and activate or deactivate the device (see Fig. 4).

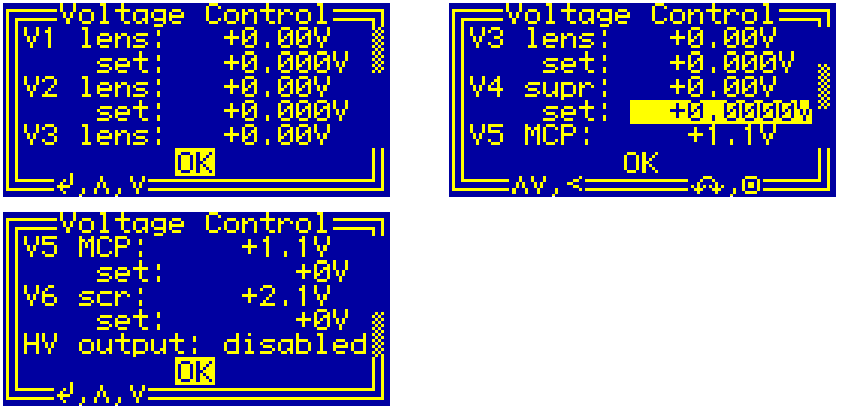

Fig. 4. Dialog box "Voltage Control".

The dialog box shows the measured values of the voltages (for more details, see section "Voltage Monitor") and their control values in the second line labeled by "set". Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the settings can by modified or the dialog box can be closed (see the selection shown at the left side of Fig. 4). The numerical values can only be modified by the rotary encoder. To be able to adjust large numbers, the voltage step may be increased by pressing and holding the direction key "left" when spinning the encoder. By pressing the encoder knob, the selected item can be reset to zero.

The item "HV output" shows the current device state (see Fig. 5). When the device is deactivated and no high-voltage can be produced, the text "disabled" is shown. When the device is activated, the LED "Activated" lights up, the high-voltages can be generated and the text "enabled" is displayed. By selecting the item "HV output" and pressing the middle key "enter", the device state can be toggled. Note that the

device can be activated only if the interlock loop is closed (see section "Interlock Setup") and no failure occurred. On failures, the device state cannot be changed and is permanently set to "disabled" (for more details, see section "Description").

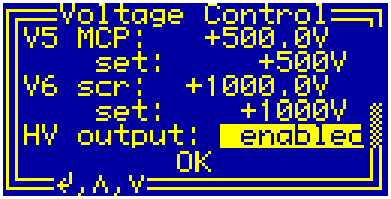

Fig. 5. Activating and deactivating the device in the dialog box "Voltage Control".

Voltage Limit Control

Using the dialog box "Limit Control", you can set the limits of the output voltages (see Fig. 6). The default values are the maximum design voltages of the respective channel. You can decrease them in order to protect the attached hardware. The settings are applied immediately and if necessary, the output voltages (see section "Voltage Control") are decreased accordingly.

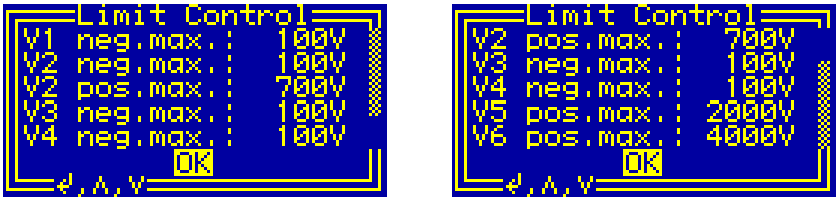

Fig. 6. Dialog box "Limit Control".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 6). The numerical values can only be modified by the rotary encoder. By pressing the encoder knob, the selected item can be set to its default value.

Voltage Slope Control

Using the dialog box "Slope Control", you can set the voltage slopes (see Fig. 7). These values define the voltage ramp that is used if the output voltage is modified (see section "Voltage Control").

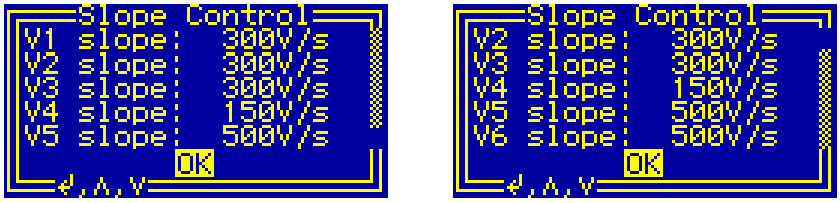

Fig. 7. Dialog box "Slope Control".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 7). The numerical values can only be modified by the rotary encoder. By pressing the encoder knob, the selected item can be set to its default value.

Amplitude Control

Using the dialog box "Amplitude Control", you can set the voltage amplitudes (see Fig. 8). These values define the voltage amplitudes that are used if the output voltage is toggled in the lock-in mode (see section "Measurement Control"). The output voltage is set either to the values defined in the dialog box "Voltage Control" (see Fig. 4) or to those values increased by the amplitude values in the dialog box "Amplitude Control" (see Fig. 8).

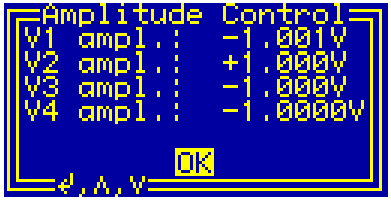

Fig. 8. Dialog box "Amplitude Control".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the dis-

played characters. By pressing the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 8). The numerical values can only be modified by the rotary encoder. By pressing the encoder knob, the selected item can be set to its default value.

Note that it is only possible to set values that can be output, i.e. that do not exceed the limits defined in the dialog box "Limit Control" (see Fig. 6).

Lock-in Control

Using the dialog box "Lock-in Control", you can set the timing of the lock-in mode (see Fig. 9). The values "Del.Trig" and "Del.Frame" define the delays between the voltage switching and the trigger pulse and between the end of the camera frame signal and the next voltage switching, respectively.

The values "Tout.Trig" and "Tout.Frame" define the timeouts waiting for the camera frame signal after a trigger pulse and for the duration of the camera frame signal, respectively.

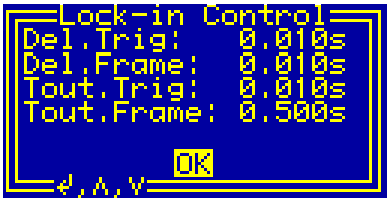

Fig. 9. Dialog box "Lock-in Control".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 9). The numerical values can only be modified by the rotary encoder. By pressing the encoder knob, the selected item can be set to its default value.

The maximum settable value for each delay or timeout is 10 s, the minimum value is 1 ms. Fig. 9 shows the default values.

Measurement Control

Using the dialog box "Measurement Control", you can control the measurement using the lock-in mode (see Fig. 10). You can set the number of frames in one measurement sequence. The device indicates the number of acquired frames as well as the current measurement state, the step within one frame, and the current output voltage.

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 10). The numerical values can only be modified by the rotary encoder. By pressing the encoder knob, the selected item can be set to its default value.

The measurement state can be "idle", "running", "stopping", or "error". The state "idle" indicates that the measurement is stopped and it can be started by the button "Run" (see Fig. 10 left). An active measurement is indicated by the state "running", it can be stopped by the button "Stop" (see Fig. 10 right). In this case, the measurement state changes to "stopping" and the button "Run" to "Term". The measurement state "stopping" indicates that the device is completing the current frame and stopping the measurement. The measurement can be stopped immediately by the button "Term", i.e. by pressing middle key "enter" again. The measurement can also be interrupted by an error; in this case, the state changes to "error" and the step indicates the condition during which the error occurred. The measurement can be restarted by the button "Run" (see Fig. 10 left).

The step within one frame changes periodically between several values (see Tab. 1).

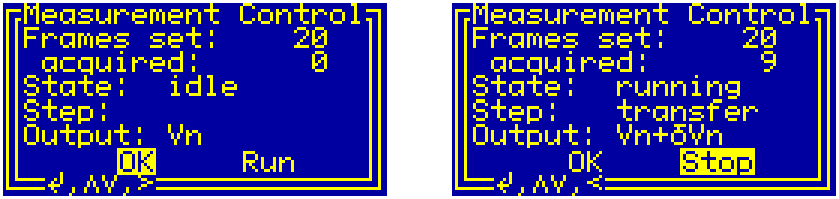

Fig. 10. Dialog box "Measurement Control".

Tab. 1. Step values of a measurement.

Step	Explanation	Error
voltage check	device tests if the voltage output is activated	voltage output is not active
ready check	device tests if the camera state signal is not active	camera state signal is active, check if it is configured properly
setup	new voltage values are being set	–
voltage set	new voltage value was set	–
triggered	trigger pulse for the camera was issued	timeout waiting for trigger response, i.e. acquisition start
acquiring	camera is acquiring data	timeout waiting for completed acquisition
transfer	camera transfers data to the control computer	–

Each measurement frame starts by checking if the voltage output is activated and if the camera state signal is not active. Then the output voltages are set to the values in the dialog box "Voltage Control" (see Fig. 4). This is indicated by the output value "Vn" and when the voltage values are reached, the step changes to "voltage set". Note that the device applies the slope values here as well (see section "Voltage Slope Control"). This may introduce a certain delay if large amplitude values are used.

When the voltage values are reached, the device introduces a delay defined by the value "Del.Trig" in the dialog box "Lock-in Control" (see Fig. 9). This is intended to allow the device to settle the output voltages. For small amplitude values (up to about 100 mV), only 5-10 ms are required, whereas for larger amplitude values, a significantly longer delay may be necessary. The proper delay has to be determined experimentally.

After the delay, a trigger pulse is output and the step changes to "triggered". When the camera has received the trigger event, the frame

signal is activated. This indicates that the camera has started the image acquisition which is shown by the step "acquiring". If this does not happen within the timeout interval "Tout.Trig" (see Fig. 9), i.e. if the camera does not respond properly, the measurement is stopped with an error message.

When the camera frame signal becomes inactive, the camera has completed the image acquisition. The acquired camera data is now transferred to the control computer and the step changes to "transfer". A timeout can also interrupt the measurement here if the camera frame signal is not deactivated within the predefined timeout interval "Tout.Frame" (see Fig. 9).

After the delay "Del.Frame" in the dialog box "Lock-in Control" (see Fig. 9), the first half of the frame is completed. The second half starts with the checks (steps "voltage check" and "ready check"), then the output voltages are set to the values defined by adding up the voltages in the dialog box "Voltage Control" (see Fig. 4) and the amplitude values in the dialog box "Amplitude Control" (see Fig. 8). This is indicated by the output value "Vn+δVn".

When the second half of the frame is completed, the number of the frames is incremented and shown as the value "acquired". The measurement ends when the number of frames reaches the set value "Frames set".

Device Setup

Camera Preferences

The dialog box "Camera Preferences" (see Fig. 11) shows the camera settings and enables you to change them.

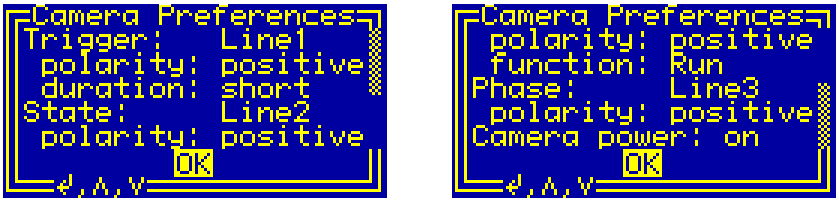

Fig. 11. Dialog box "Camera Preferences".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the settings can by modified or the dialog box can be closed (see the selection shown in Fig. 12). The numerical values can only be modified by the rotary encoder. By pressing its knob, the selected item can be reset to its default value.

The value "Trigger" defines the hardware signal that is used to send the trigger signal to the camera. It can be changed between "Line1" and "Line3". The next item "polarity" defines the active signal level, it can be set to "positive" or "negative". The field "duration" defines the length of the trigger pulse. If set to "short", an approximately 1 ms long trigger pulse is generated. The other possible value is "adaptive", in this case, the trigger pulse ends when the camera signal "State" becomes active.

The value "State" is the hardware signal that is used to receive the camera state signal from the camera. It can be changed between "Line2" and "Line3". The item "polarity" defines the active signal level, "function" determines its functionality. If set to "Run", the signal is expected to indicate that the camera is acquiring data. In this case, the device applies the timeout value "Tout.Trig" in the dialog box "Lock-in Control" (see Fig. 9). The timeout starts with the trigger signal and if the signal "State" is not activated within the set timeout, an error occurs and the measurement stops (see Tab. 1). The timeout value

"Tout.Frame" in the dialog box "Lock-in Control" (see Fig. 9) defines the maximum length of the signal "State". As above, an error occurs and the measurement stops in the event of a timeout.

If the item "function" of the signal "State" is set to "Ready", the camera reports via the active signal "State" that the data acquisition is completed and the camera is ready to receive the next trigger signal. In this case, the timeout value "Tout.Trig" in the dialog box "Lock-in Control" (see Fig. 9) does not have any function. It is expected that the camera immediately receives the trigger signal and that the signal "State" must be activated within the timeout value "Tout.Frame" in the dialog box "Lock-in Control" (see Fig. 9). As above, an error occurs and the measurement stops in the event of a timeout.

The last signal "Phase" is optional. It can be set to "Line1" or "Line3" or it can be disabled (see Fig. 11 right). If enabled, it outputs the phase state. When "polarity" is set to "positive", it outputs a logical 0 if the output voltages are set to the values in the dialog box "Voltage Control" (see Fig. 4), and a logical 1 if the output voltages are set to the values defined by adding up the voltages in the dialog box "Voltage Control" (see Fig. 4) and the amplitude values in the dialog box "Amplitude Control" (see Fig. 8). For more details, see section "Measurement Control".

Fan Preferences

The dialog box "Fan Preferences" (see Fig. 12) shows the current fan settings and enables you to change them.

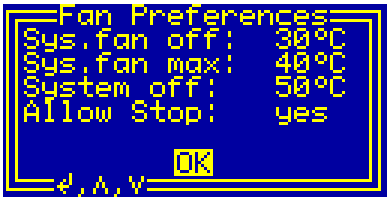

Fig. 12. Dialog box "Fan Preferences".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the settings can by modified or the dialog box can be closed (see the selection shown in Fig. 12). The numerical values can only be modified by the

rotary encoder. By pressing its knob, the selected item can be reset to its default value.

The values "Sys.fan off" and "Sys.fan max" define the temperature thresholds for the fan controller. If the temperature measured by the attached sensors is lower than the lower threshold "Sys.fan off", the fan rotates at its minimum speed. On the other hand, it rotates at its maximum speed if the measured temperature reaches the upper threshold "Sys.fan max".

The fan can be turned off if the item "Allow Stop" is set to "yes". The hysteresis is 1°C, i.e. the fan starts to rotate if the temperature exceeds the lower threshold but stops at a temperature lower by 1°C.

The function of the fan can be checked using the dialog box "Fan Monitor" (see section "Fan Monitor").

If the temperature exceeds the shutdown threshold "System off", the device is deactivated and overheating is signalized. The device can be activated again once the temperature has decreased below the shutdown threshold, the hysteresis is 1°C.

The threshold values can be changed using the rotary encoder. The minimum settable value is 20°C, the maximum is 60°C. The differences between the values must be at least 1°C. You can lower the difference between the temperature thresholds "Sys.fan off" and "Sys.fan max" if the device should stabilize its internal temperature. The proper values have to be found experimentally, since they depend on the room temperature and on the output power of the device.

Interlock Setup

The dialog box "Interlock Setup" (see Fig. 13) shows the current interlock settings and enables you to change them.

Fig. 13. Setting the interlock function in the dialog box "Interlock Setup".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. By pressing the middle key "enter", the settings can by modified or the dialog box can be closed (see the selection shown in Fig. 13).

If the selected value is enabled, the state of the corresponding interlock loop is evaluated. The device's interlock loop is assumed to be closed and the device can be activated, if all enabled loops are closed. The state of the interlock loop is signalized by the yellow LED "Interlock", the state of each particular loop can be displayed in the dialog box "Interlock Monitor" (see section "Interlock Monitor"). Note that if all loops are disabled, the LED "Interlock" lights permanently and the device can be enabled independently on the physical state of the interlock loops.

The contacts of the loops "Front BNC" and "Rear BNC" are located in the BNC sockets at the front and rear panel, respectively. The particular loop can be closed by shorting the BNC socket, i.e. by connecting its signal to device ground.

Interface Setup

The dialog box "Interface Setup" (see Fig. 14) shows the current setting of the USB interface and allows you to change it.

Fig. 14. Dialog box "Interface Setup".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. The settings can by modified by pressing the middle key "enter". Furthermore, the dialog box can be closed using the middle key "enter" (see the selection shown in Fig. 14).

The value "USB port" determines whether the USB interface is active or not (see Fig. 15). By pressing the middle key "enter", the value can

be toggled between "active" or "idle" and "off". If the interface state is "off", the interface is disabled and no communication is possible. The remaining interface states indicate that the interface is enabled. If the displayed state is "active", the interface is activated and a communication can be launched. If the displayed state is "idle", the interface is activated but does not communicate, i.e. there is no connection to a remote host. If a cable connects the device with a functioning remote host, the state changes to "active".

Fig. 15. Enabling the USB interface in the dialog box "Interface Setup".

The communication state of the USB interface is shown as the item "comm.", it indicates the state of the handshake lines of the interface. The communication state changes between "active" and "idle" depending on the communication activity on the interface. The communication state is "idle" if you do not access the device and it changes to "active" if you start a communication software on the remote host.

Display Setup

The dialog box "Display Setup" (see Fig. 16) shows the current settings of the display controller and allows you to change them.

Fig. 16. Dialog box "Display Setup".

Use the vertical direction keys to change the currently selected item. The selected item is highlighted by inverting the colors of the displayed characters. The settings can by modified by pressing the mid-

dle key "enter" or by spinning the rotary encoder. Furthermore, also using the middle key "enter", the dialog box can be closed (see the selection shown in Fig. 16). The numerical values can only be modified by the rotary encoder. By pressing its knob, the selected item can be reset to its default value.

Fig. 17. Setting the brightness in the dialog box "Display Setup".

The value "Brightness" shows the luminosity of the display, it can be changed by spinning the encoder (see Fig. 17). By pressing its knob, the luminosity is reset to its default value. Note that the lifetime of the display is inversely proportional to the brightness, thus you should chose the lowest possible value that provides a sufficient luminosity.

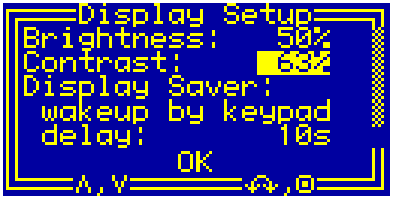

Fig. 18. Setting the contrast in the dialog box "Display Setup".

The value "Contrast" determines the display contrast, it can be changed by spinning the encoder (see Fig. 18). By pressing its knob, the contrast is reset to its default value. By changing this value, the readability of the display for a certain view angle can be optimized.

Fig. 19. Setting the display saver in the dialog box "Display Setup".

The value "Display Saver" determines the functionality of the display saver, it can be changed by pressing the middle key "enter" (see Fig. 19). If disabled, the display is permanently lit up with the luminosity given by the value "Brightness" (see Fig. 17). In both remaining cases, the luminosity is reduced after a predefined inactivity time in order to increase the display lifetime. If "wakeup by keypad" is selected, the display saver is stopped by pressing any key or operating the encoder. The luminosity given by the value "Brightness" is restored and the device is ready for further operation. If "wakeup by all" is selected, the display saver is also stopped by pressing a virtual key using the remote controller (see section "Utility RemoteControl").

Fig. 20. Setting the display saver delay in the dialog box "Display Setup".

The value "delay" defines the delay of the display saver, it can be changed by spinning the encoder (see Fig. 20). By pressing its knob, the delay is reset to its default value. The delay value is the inactivity time required to launch the display saver, the allowable values are between 3 and 9999 seconds. Be sure that you set the delay to a sufficiently high value that will not disturb normal operation. On the other hand, a too large value will completely prevent the display saver from launching, thus a compromise has be found for practical usage.

Fig. 21. Setting the display saver brightness in the dialog box "Display Setup".

The value "brightness" determines the luminosity of the display when the display saver is active, it can be changed by spinning the encoder (see Fig. 21). By pressing its knob, the brightness is reset to its default value. It should be set to lowest possible value that provides a minimum luminosity indicating that the device is still running.

Note that the display luminosity is changed to the value "brightness" if the item is selected. If a very low luminosity value is set, the display may become unreadable. In such case, either press the encoder knob to reset the brightness to its default value or use the remote controller (see section "Utility RemoteControl") for adjusting the display settings.

Device Monitoring

To monitor the function of the device, several dialog boxes are available. You can use them to check the function of the device if you encounter any problem, for instance, if the red LED "Failure" lights.

Voltage Monitor

The dialog box "Voltage Monitor" (see Fig. 22) shows the current values of the output high voltages.

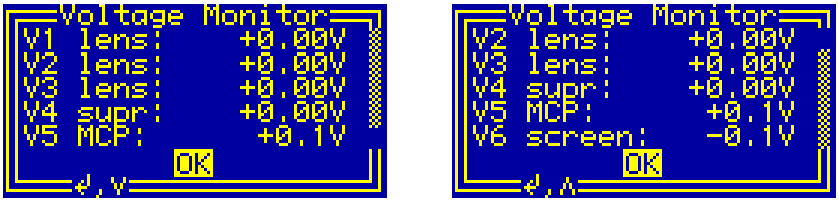

Fig. 22. Dialog box "Voltage Monitor".

Use the vertical direction keys to select the shown items. By pressing the middle key "enter", the dialog box can be closed.

Under normal conditions when the device is activated, the measured voltages should be equal to the voltage levels set by the dialog box "Voltage Control" (see section "Voltage Control"). The values V5 and V6 should have positive, the voltages V1, V3, and V4 negative values, the voltages V2 can show values of both polarities. If the device is deactivated and the high voltages are disabled, also small voltages with an opposite polarity may be shown.

If the measured voltages differ from the set values, the device may be overloaded or overheated or a hardware failure occurred. You may try to localize the failure by checking the device parameters by observing other monitoring dialog boxes. You may also try to disconnect the output cables and repeat the last operation in order to exclude the influence of the connected experimental setup. If the issue persists, turn off the device and contact the manufacturer.

The values are measured about 4 times per second by a bipolar 16-bit analog-to-digital converter. The resolution depends on the voltage range of the particular channel. The voltages V1, V3, and V4 have a

resolution of about 5 mV, the voltage V2 only 30 mV. The high voltages V5 and V6 have a fixed resolution of 0.1 V.

Note that the voltage measurement is also influenced by the voltage at the analog ground (see section "Supply Monitor"). If the voltage at the analog ground is ±2 V or more, the measurement of the output voltages provides wrong values.

If you aim to manipulate the output cables, the connectors, or the vacuum hardware, use the dialog box "Voltage Monitor" to check the high voltages. Be sure that you open the connectors or touch the vacuum hardware only if the high voltages have decreased to values that are not lethal.

Supply Monitor

The dialog box "Supply Monitor" (see Fig. 23) shows the current values of the supply voltages of the device.

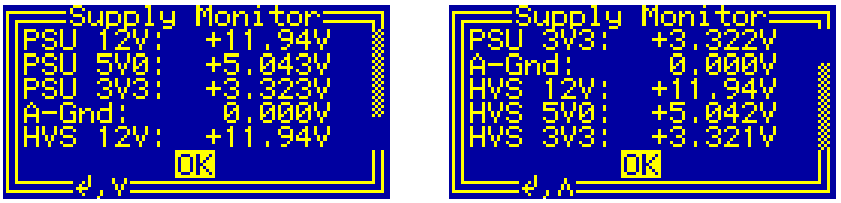

Fig. 23. Dialog box "Supply Monitor".

Use the vertical direction keys to select the shown items. By pressing the middle key "enter", the dialog box can be closed.

The value "PSU 12V" is the internal voltage supplying the fan and the high-voltage converters generating the output voltages V5 and V6. The nominal value is 12 V with a tolerance of 2% maximum.

The value "PSU 5V0" is the internal voltage supplying the digital circuits of the device including the LCD. The nominal value is 5 V with a tolerance of 1% maximum.

The value "PSU 3V3" is the internal voltage supplying the fast digital circuits. The nominal value is 3.3 V with a tolerance of 1% maximum.

The value "A-Gnd" is the voltage at the analog ground. The analog ground should be connected to the digital ground, i.e. to the protection earth (PE) by the attached experimental setup. Typical experimental

setups are grounded by the used vacuum pumps or other hardware, thus are connected to the protection earth (PE) and the analog-ground wire in the output cable should just be connected to a metallic part of the experimental setup to provide the proper grounding of the device. Due to voltage drops in connection cables, a nonzero voltage may be indicated. However, voltages larger than 100 mV most probably indicate an issue with the cables or connectors that must be solved before using the device.

Please note that the range of the voltage measurement at the analog ground is about ±2 V only. If such values are displayed, also the measurement of the output voltages (see section "Voltage Monitor") may be corrupted by the overloaded measurement of the analog-ground voltage.

The values "HVS 12V", "HVS 5V0", and "HVS 3V3" should be almost identical to their counterparts "PSU 12V", "PSU 5V0", and "PSU 3V3". They are measured by the module of the high-voltage converters HV-SMPS2. Any difference larger than several 10 mV indicates a malfunction of the device. If you observe such failure, turn off the device and contact the manufacturer.

The values are measured about 2 times per second. The microcontroller checks them continuously and deactivates the device if the voltages differ from their nominal values. This may happen when the device is overloaded, the supply voltage is shorted, or when one of the power supplies overheats.

Interlock Monitor

The dialog box "Interlock Monitor" (see Fig. 24) shows the state of the interlock loops.

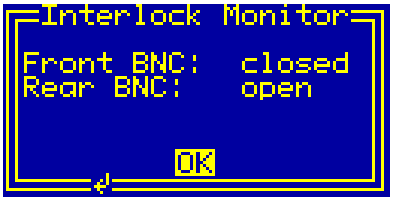

Fig. 24. Dialog box "Interlock Monitor".

For the assignments, consult the dialog box "Interlock Setup" (see section "Interlock Setup"). The possible states are "open" and "closed", they correspond directly to the state of each particular loop.

Note that the dialog box "Interlock Monitor" shows the real state of the particular loop, independently on that if it is enabled or not. If a certain interlock loop cannot be closed due to a malfunction, you may deactivate it in the dialog box "Interlock Setup" to be able to continue the experimental work.

Temperature Monitor

The dialog box "Temperature Monitor" (see Fig. 25) shows the temperatures measured in different parts of the device.

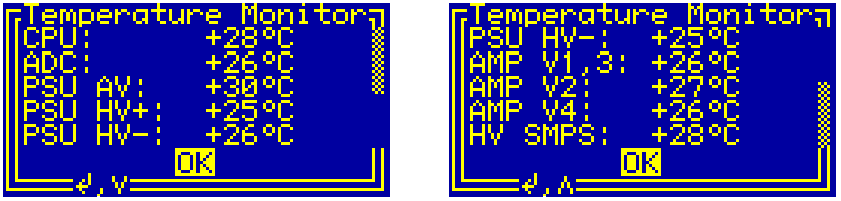

Fig. 25. Dialog box "Temperature Monitor".

Use the vertical direction keys to select the shown items. By pressing the middle key "enter", the dialog box can be closed.

The value "CPU" is the chip temperature of the controller's CPU. Under normal conditions, i.e. an ambient temperature between 20 and 25°C, it approaches values between 25 and 30°C.

The value "ADC" is the chip temperature of an analog-to-digital converter (ADC) located in the controller in the proximity of the rear connector. Under normal conditions, the value stabilizes between 25 and 30°C.

The values starting with "PSU" are the heatsink temperature of the device's power supplies. The value "PSU AV" indicates the temperature of the analog power supply of ±15 V, the "PSU HV+" of the positive high-voltage power supply of about +720 V, and the "PSU HV-" of the negative high-voltage power supply of -120 V, respectively. Under normal conditions, these temperature approaches values around 30°C if the device is activated.

The values "AMP" are the heatsink temperature of the amplifiers producing the adjustable output voltages V1 - V4. The value "AMP V1,3" indicates the temperature of the amplifier HVA100-2ND producing the negative voltages V1 and V3. The values "AMP V2" and the "AMP V4" are the temperature of the amplifiers HVA700-1DH and HVA100-1NDH generating the voltages V2 and V4, respectively. Under normal conditions, these temperature approaches values around 30°C if the device is enabled.

The item "HV SMPS" is the chip temperature of the CPU in the module of the high-voltage converters HV-SMPS2. Under normal conditions, the value stabilizes between 25 and 30°C.

The values are measured about 2 times per second. The microcontroller checks the values continuously, controls the fans, and deactivates the device if the temperature rises above the allowed values (see sections "Fan Preferences" and "Fan Monitor" for more details). The microcontroller is also able to discover failures such as a disconnected or shorted sensor; any failure causes an immediate deactivation of the device.

Fan Monitor

The dialog box "Fan Monitor" (see Fig. 26) shows the state of the fan mounted at the rear panel of the device.

Fig. 26. Dialog box "Fan Monitor".

The value "Rear fan" shows the rotational speed of the fan. The fan speed is shown in revolutions per minute (rpm). The possible range of the fan speed is given by the used fan, please consult the manual of the main device for the speed values.

The value "state" indicates the state of the fan. The values shown may be either "running" if the fan is running properly, "stopped" if the fan is halted, or "failed" if the fan has failed and does not rotate. The state

"failed" may be shown for a short time after the fan start, this is a normal behavior since the fan needs several seconds to run up. If the failure persists for a longer time (about 10 s), the red LED "Failure" lights up. Note that this does not prevent the device from operation but an overheating may occur sooner. You should replace the defective fan as soon as possible to maintain proper device operation.

Camera Monitor

The dialog box "Camera Monitor" (see Fig. 27) shows the current logical levels of the camera signals.

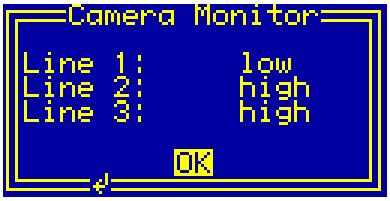

Fig. 27. Dialog box "Camera Monitor".

The values correspond to the respective camera signal ("Line 1", "Line 2", and "Line 3"). The possible values are "low" for logical 0 and "high" for logical 1. You can use this dialog box to check whether the camera receives the correct signals and provides the expected feedback.

Clock Monitor

The dialog box "Clock Monitor" (see Fig. 28) shows the current frequencies of various clock signals used internally to control the device.

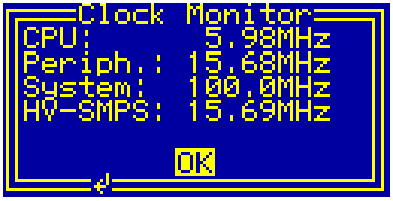

Fig. 28. Dialog box "Clock Monitor".

The value "CPU" is the clock frequency of the main microcontroller that controls the device. Since the microcontroller clock is dynamically adjusted according to the current CPU load, this value may vary according to the number and the complexity of active processes.

The value "Periph." is the frequency used to control the peripheral devices of the microcontroller, the microcontroller clock is derived from this signal. It is regulated by the microcontroller to a value of about 15.67 MHz.

The value "System" is the frequency of the main clock controlling the FPGA (*Field Programmable Gate Array*) in the controller. It is gained by a PLL (*Phase Locking Loop*) circuitry. The PLL multiplies the frequency of a quartz oscillator and provides a clock of nominally 100 MHz. This clock is internally used to control the communication signals or to generate the PWM (*Pulse-Width Modulation*) control signal for the fan.

The value "HV-SMPS" is similar to the value "Periph.", it is the frequency used to control the peripheral devices of the microcontroller in the high-voltage converters HV-SMPS2. It is regulated by this microcontroller to a value of about 15.67 MHz.

The frequencies are measured as ratios between the particular frequency and the frequency of an auxiliary quartz oscillator running at 32.768 kHz. If this oscillator fails, all measured values are subjected to systematic deviations.

Line Monitor

The dialog box "Line Monitor" (see Fig. 29) shows the current state of the line input.

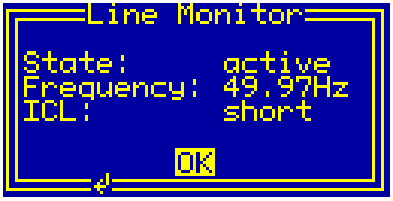

Fig. 29. Dialog box "Line Monitor".

The value "State" indicates the state of the line input. The value "active" is shown if the line voltage is present. If the line voltage is absent,

the state changes to "down" and the device shuts down after a short time period.

The value "Frequency" is the line frequency measured by the device. Dependent on the location, the nominal values are 50 or 60 Hz. Due to natural deviations, the line frequency may slightly differ from the nominal value.

The value "ICL" shows the state of the inrush current limiter (ICL). The ICL is activated when the device starts and the indicated ICL state is "active". After a few seconds when the large initial charging current from the line input drops down to the nominal operational current, the ICL is deactivated, i.e. shorted and the indicated ICL state is "short".

Device State

To inspect the device state, several dialog boxes are available.

System Information

The dialog box "System Information" (see Fig. 30) summarizes general information about the device controller.

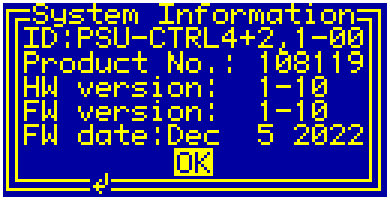

Fig. 30. Dialog box "System Information".

The text "ID" shows the product identification, "Product No." is the unique serial number of the product. The entries "HW version" and "FW version" are the hardware and the firmware versions, "FW date" is the compilation date of the firmware.

In case of any trouble with the device, please note these values, save the device parameters (see section "Utility COM-HVPSU6-Test"), or make a hardcopy of this dialog box (see section "Utility RemoteControl") before contacting the manufacturer.

HV-SMPS2 Information

The dialog box "HV-SMPS2 Information" (see Fig. 31) summarizes general information about the module of the high-voltage converters HV-SMPS2.

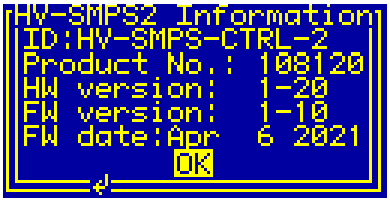

Fig. 31. Dialog box "HV-SMPS2 Information".

The shown items are similar to those of the dialog box "System Information" (see Fig. 30): The text "ID" shows the module identification, "Product No." is the unique serial number of the product. The entries "HW version" and "FW version" are the hardware and the firmware versions, "FW date" is the compilation date of the firmware.

In case of any trouble with this module, please note these values, save the device parameters (see section "Utility COM-HVPSU6-Test"), or make a hardcopy of this dialog box (see section "Utility RemoteControl") before contacting the manufacturer.

Statistics

The dialog box "Statistics" (see Fig. 32) shows information about the operation time of the device. The "Uptime" indicates the time elapsed since powering on the device or since the last firmware restart. The next line "total" shows the total operation time of the device.

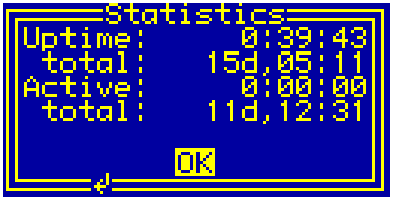

Fig. 32. Dialog box "Statistics".

The time "Active" displays the operation time of the high-voltage power supplies and the high-voltage amplifiers and converters, which has elapsed since powering on the device or since the last firmware restart. The next line "total" shows the total operation time of the device.

Tab. 2. The LCD character set.

value	char	value	char	value	char	value	char
20h	space	38h	8	50h	P	68h	h
21h	!	39h	9	51h	Q	69h	i
22h	"	3Ah	:	52h	R	6Ah	j
23h	#	3Bh	;	53h	S	6Bh	k
24h	$	3Ch	<	54h	T	6Ch	l
25h	%	3Dh	=	55h	U	6Dh	m
26h	&	3Eh	>	56h	V	6Eh	n
27h	'	3Fh	?	57h	W	6Fh	o
28h	(	40h	@	58h	X	70h	p
29h	)	41h	A	59h	Y	71h	q
2Ah	*	42h	B	5Ah	Z	72h	r
2Bh	+	43h	C	5Bh	[	73h	s
2Ch	,	44h	D	5Ch	\	74h	t
2Dh	–	45h	E	5Dh	]	75h	u
2Eh	.	46h	F	5Eh	^	76h	v
2Fh	/	47h	G	5Fh	_	77h	w
30h	0	48h	H	60h	`	78h	x
31h	1	49h	I	61h	a	79h	y
32h	2	4Ah	J	62h	b	7Ah	z
33h	3	4Bh	K	63h	c	7Bh	{
34h	4	4Ch	L	64h	d	7Ch	|
35h	5	4Dh	M	65h	e	7Dh	}
36h	6	4Eh	N	66h	f	7Eh	~
37h	7	4Fh	O	67h	g	7Fh	©

Software Utilities

The software utilities can be found in the directory "Program" of the enclosed software package, the 64-bit versions are located in the subdirectory "x64". Before using them, the virtual USB port driver must be installed (see section "Driver Installation"). The utilities do not require any additional installation, you need only to copy them to a suitable directory on your computer. Before starting the utilities, you need to obtain the virtual port number, as described in the section "Driver Installation".

Utility COM-HVPSU6-Test

The **COM-HVPSU6-Test** is a Windows™ program that runs in text mode[§]. It enables you to control and monitor the voltage controller. Launching the utility **COM-HVPSU6-Test.exe** without any parameters or with the parameter **-?** displays a simple help:

```
COM-HVPSU6-Test -?
```

To start the program without any error message, at least the number of the COM port must be given:

```
COM-HVPSU6-Test 6
```

This command starts the utility **COM-HVPSU6-Test** and assumes that the device is connected to the port COM6. On success, the utility reports the following message:

```
Press '?' for help
```

and waits for command input.

In case of any problem, check whether the port number matches the system settings (see section "Driver Installation") and the connected device is powered on and working properly (see section "Interface Setup"). If an error occurs, please consult the section "Error Codes".

[§] To run a text-mode application, select "Run" in the start menu of Windows™ and type "cmd". Then change the directory to that with the program files using the command "cd". Finally, execute the given command by copy & pasting and pressing "Enter". A better and more comfortable alternative to the Windows™ command shell are utilities like "File Commander/W" or "File and archive manager (FAR)". Please use you favorite search utility to find out how to obtain these applications.

Tab. 3. Command-line parameters of the program COM-HVPSU6-Test - general commands.

Parameter	Explanation
-b, -B	get the device buffer status
-z, -Z	purge the communication
-t, -T	terminate the program
-q, -Q	quiet mode
-g	debug mode
-G	debug mode with output into **Debug.txt**
-#	restart the device
-?	show the online help

The tables 8 and 9 explain the possible error messages; they should help you to localize the reason for the software failure.

To check the communication, press the key 'p' to obtain the product identification text. The device should respond as follows:

Product identification: PSU-CTRL4+2,1-00

If the device responds properly, you may try other program commands. Press '?' to obtain the help listing of all available commands[**].

In practice, you may prefer to use the command-line mode instead of the interactive mode. The command-line mode allows you, for instance, to save complete commands in batch files for repeated usage.

General commands

Table 3 summarizes general command-line parameters of the program **COM-HVPSU6-Test**. The parameters are processed from left to right. If an error in the command line is encountered, the program stops with a text describing the error.

If the parameter **-t** is found, the program stops without processing any following parameter. If you do not specify the parameter **-t** at all,

[**] Note that keyboard layouts different to the US one may cause issues when evaluating several characters. We recommend to switch to the US layout when using the utility COM-HVPSU6-Test in the interactive mode.

Tab. 4. Command-line parameters of the program COM-HVPSU6-Test - getting device parameters.

Parameter	Explanation
-V	get device firmware version
-v	get device hardware version
-d, -D	get device firmware date
-p, -P	get product identification text
-n, -N	get product number

the program does not stop and enters the interactive mode after having processed the complete command line.

The quiet program mode is turned on by the parameter **-q**. If specified, the program text output is reduced. In contrary to that, the debug mode (parameters **-g** or **-G**) provides a detailed output for error analysis. The debug mode is intended to be used in case of communication problems. The parameter **-G** creates additionally the file **Debug.txt** in the current directory, the file contains a detailed logging of the communication. In case of communication issues, turn on the debug mode by this parameter and send the file to the manufacturer if you cannot resolve the problems.

The parameters **-z** or **-b** can be used if the communication does not work properly. The parameters **-z** or **-Z** clear the communication buffers and restarts the communication. The commands **-b** or **-B** show the device buffer status, it should be empty since the device should process all commands immediately. If the device buffer contains any data, most probably, a communication error occurred and the next command will not be processed properly.

Getting device parameters

The command-line parameters for obtaining the device parameters are listed in Tab. 4. You can use them to identify the device and check the firmware version. For more details, see section "Device State". To collect a complete set of device information, consult section "Device testing".

Tab. 5. Command-line parameters of the program COM-HVPSU6-Test - device monitoring.

Parameter	Explanation
-u	get device uptime
-U	get device uptime periodically
-c	get CPU data
-C	get CPU data periodically
-f	get HV-SMPS2 CPU data
-F	get HV-SMPS2 CPU data periodically
-h	get device housekeeping data
-H	get device housekeeping data periodically
-k	get HV-SMPS2 housekeeping data
-K	get HV-SMPS2 housekeeping data periodically
-s	get device state
-S	get device state periodically
-j	get measured temperatures
-J	get measured temperatures periodically
-r	get measured voltages
-R	get measured voltages periodically

Device monitoring

Table 5 shows command-line parameters for monitoring the device. They are useful in case of a malfunction when you are searching for the reason of unwanted issues. For more details, see section "Device Monitoring". To measure a complete set of device data, consult section "Device testing".

Device control

The command-line parameters for controlling the device are shown in Tab. 6. You can use them to activate the device or control the output voltages. For more details, see section "Device Control".

Tab. 6. Command-line parameters of the program COM-HVPSU6-Test - device control.

Parameter	Explanation
-a	turn the HV off
-A	turn the HV on
-mn	get the voltage limit of the channel number **n**
-Mn Limit	set the voltage limit of the channel number **n** to the value **Limit**
-on	get the voltage slope of the channel number **n**
-On Slope	set the voltage slope of the channel number **n** to the value **Slope**
-ln	get the voltage of the channel number **n**
-Ln Voltage	set the voltage of the channel number **n** to the value **Voltage**

The device can be activated by the parameter **-A** and deactivated by the parameter **-a**. After issuing this command, the utility scans the device state periodically. You can check the device state and determine whether the activating was successful.

To obtain the currently set voltage limit, the parameter **-m** can be used. The parameter requires a numerical value **n** that specifies the channel. Since the channel V2 has two limits (first for the negative and second for the positive voltages), the channel assignment is as follows: 0 = V1, 1 = V2 negative, 2 = V2 positive, 3 = V3, .., 6 = V6. Using the parameter **-M**, the voltage limits can be set. Beside the numerical value **n** specifying the channel, the parameter requires the new value of the voltage limit given in volts.

Use the parameter **-o** to obtain the currently set the voltage slope. The parameter requires a numerical value **n** that specifies the channel. The channel assignment is as follows: 0 = V1, 1 = V2, .., 5 = V6. By the parameter **-O**, the voltage slopes can be set. Beside the numerical value **n** specifying the channel, the parameter requires the new value of the voltage slope given in volts per second.

Tab. 7. Command-line parameters of the program COM-HVPSU6-Test - device testing.

Parameter	Explanation
-i FileName	save device data to a text file with the name **FileName**
-I FileName	log device data to a text file with the name **FileName**
-x FileName **-X FileName**	create test protocol to a text file with the name **FileName**
-e FileName n s p d t **-E FileName n s p d t**	scan output voltages and save them to a text file with the name **FileName**. **n** is the channel number, **s** the starting voltage, **p** the final voltage, **d** the voltage step, and **t** the delay before a measurement for stabilizing the voltage

The output voltages can be shown by the parameter **-l**. The numerical value **n** specifies the channel, the assignment is identical to the previous command: 0 = V1, 1 = V2, .., 5 = V6. To set the output voltages, use the parameter **-L**. The second numerical value that must be specified is the voltage given in volts.

Device testing

Table 7 summarizes command-line parameters of the program **COM-HVPSU6-Test** for testing the device. They can be used to create log files or measurement protocols automatically.

Note: If you wish to specify a name parameter containing spaces or special characters, use the conventions valid for your operating system. In Windows™ systems, for instance, enclose the name in quotation marks.

Using the parameter **-i**, a summary of device data can be stored in a specified text file. By the parameter **-I**, a log file with all measured device values is created. The values are obtained 2 times per second and stored in a tab-separated text file. The logging can be stop by pressing the 'Esc' key. The output file can be opened in any text editor and easily copied in any spreadsheet software.

The parameters **-x** or **-X** can be used to create a test protocol of the device. You can start this function in order to test the device and compare the resulting protocol with that one that was created by the manufacturer.

Utility FlashLoader

The FlashLoader is a simple 32-bit Windows™ program running in text mode (see note "text mode" for more information). It enables you to upgrade the firmware of the voltage controller. You should perform the upgrade if you have received or downloaded a new firmware file from the device manufacturer. Launching the utility **FlashLoader.exe** without any parameters displays a simple help text with the expected syntax of the command line.

Before upgrading the firmware, you should first test the device and the communication by verifying the current firmware version. To do so, start the following command:

```
FlashLoader 6 Firmware.txt -v
```

where **Firmware.txt** is the file containing the current firmware and the number 6 indicates the port COM6 to which the device is connected. The program should produce the following output:

```
Code file R:Final.txt from 10/06/2020, 12:00:00
Flash Loader 1.12
Verifying code file Final.txt
Verifying finished at Tue, 10/06/2020, 15:00:00
100495 (1888Fh) bytes processed, 99584 (18500h)
bytes verified
Resetting the target
Program finished ok
```

During the verify procedure, a message box is shown at the device display informing the user that the FlashLoader has been activated. When the verify finishes without any error, the device is restarted.

Note that the utility FlashLoader cannot communicate with the device if it is not in the idle state (see Fig. 2).

! **Attention:** To be sure that the device cannot apply high voltages to the attached experimental setup when the FlashLoader is active, disconnect the output cables. Note that the utility stops all processes running at the microcontroller, thus the monitoring is not available and the device would not be deactivated if a critical situation arises.

If any error occurs, do not proceed with the firmware upgrade. If you cannot resolve the issues, contact the manufacturer. Note that even if the verify fails and the flash loader at the device remains active, it is

safe to power the device off to restart it. However, a more safe and comfortable alternative to that is to execute the following command:

```
FlashLoader 6 -i -f
```

This prevents the utility at the host computer from initializing the flash loader utility at the microcontroller again and sends the reset command to the device.

If the verify has succeeded, you may start the firmware upgrade by entering the command:

```
FlashLoader 6 Firmware.txt
```

where **Firmware.txt** is the file with the new firmware. The program should produce the following output:

```
Code file R:Final.txt from 10/06/2020, 12:00:00
Flash Loader 1.12
Programming code file Final.txt
Programming finished at Tue, 10/06/2020, 15:00:00
100495 (1888Fh) bytes processed, 99584 (18500h)
bytes programmed
Resetting the target
Program finished ok
```

During the programming procedure, a message box is displayed at the device display. When the programming finishes, the device is restarted with the new firmware.

If an error occurs, the flash loader utility at the microcontroller may remain active. This is the case if the message box at the device display is still present and the device did not restart. In this case, you may retry the action with the command-line parameter **-i**:

```
FlashLoader 6 Firmware.txt -i
```

This will prevent the utility at the host computer from initializing the flash loader utility at the microcontroller again and it will just try to reprogram the file **Firmware.txt**. If the error persists, contact the manufacturer.

! **Attention:** You must not power down the device if the firmware upgrade did not succeed. Otherwise, the device will not operate properly or might even not restart at all. Did this happen, it would be necessary to reprogram the device in the factory.

If the current firmware is damaged so that the device is inoperable, you may try to start the flash-loader utility at the microcontroller manually. Press the horizontal direction keys (left and right) simultaneously and power on the device. If this small part of the firmware is still working, the device's flash loader will start - you can recognize it by the missing display activity after the startup. Then, try to launch the utility FlashLoader with the command-line parameter **`-s`**:

```
FlashLoader 6 Firmware.txt -s
```

This does not start the flash-loader utility at the microcontroller but will only try to reprogram the file **`Firmware.txt`**. If an error occurs that you cannot solve, contact the manufacturer.

Utility RemoteControl

The RemoteControl is a Windows™ GUI program that enables you to control the voltage controller remotely and make hardcopies of the current LCD content for documentation purposes.

To launch the utility, enter the following command at the command prompt (see note "text mode" for more information):

```
RemoteControl 6
```

The number 6 indicates the port COM6 to which the device is connected. Instead of using the command prompt, you can create a shortcut to the utility and save the port number there.

The program opens a dialog box that shows the current content of the device's LCD (see Fig. 33). Above the display area, the device identi-

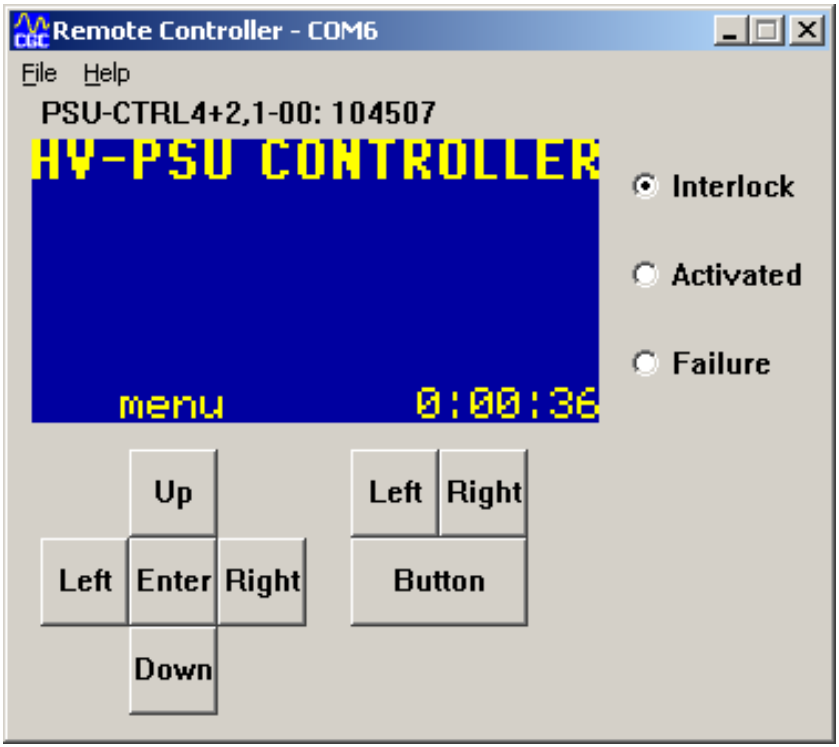

Fig. 33. Utility RemoteControl.

fication and, after a colon, the product number are shown (see also section "System Information").

The direction keys, the middle key (↵), and the encoder can be simulated by clicking the corresponding buttons or using the keyboard shortcuts. For the detailed description of them, see the online help of the utility.

The right side of the program window shows the status of the device LEDs. This allows you to observe the LED activity without the necessity of a direct visual contact to the front panel of the device.

You may obtain a hardcopy of the LCD at any time by selecting the menu item File ► Save Hardcopy. The program opens a dialog box to select the destination bitmap file. The hardcopy is performed immediately without waiting for any confirmation.

Note that the utility RemoteControl continuously transfers display data over USB. Compared to the idle state, this leads to a higher load of the microcontroller controlling the device. Further, the utility prevents any other program from accessing the device via USB.

Driver Installation

Installation of the Virtual Port for the USB Interface

The virtual port driver is required for the operation of the device with a USB interface. If you use the operating system Windows™, please note the following:

- Please use the update function of the operating system at the host computer or download the most recent driver from the homepage of the manufacturer of the USB adapter. The drivers are located at the following address: http://www.ftdichip.com/Drivers/VCP.htm. Please choose the correct driver version according to your operating system.
- To install the driver, administrative rights are required.
- The installation is described in detail in the "Installation Guides" available at the abovementioned address. Please read this description carefully before starting the installation.
- After the installation, the number of the virtual port can be adjusted. You can change the settings in the device manager by opening the settings of the device *USB Serial Port (COMx)*. To modify the settings, administrative rights are required. The settings are applied immediately, you do not need to reboot the PC to activate them.

The software can also be used at computers running the Linux operating system. You can run them using the Windows™ emulator wine (see http://www.winehq.org/).

Starting with Linux Kernel 3.0.0-19, all FTDI devices are already supported without the necessity of compiling additional kernel modules. For more details, consult the homepage of the manufacturer of the USB adapter: http://www.ftdichip.com/Drivers/VCP.htm.

The system has to be configured an the following way:

- Use, for instance, the program **`dmesg`** to find out to which USB port the device is attached: Look for a line similar to "FTDI USB Serial Device converter now attached to ttyUSB0".
- Link the Linux device to the virtual COM port of wine:
  `ln -s /dev/ttyUSB0 ~ /.wine/dosdevices/com3`
  This assumes that the device is attached to ttyUSB0 and will be linked with COM3.

Software Interface

The software interface of the device consists of a 32-bit dynamic link library **COM-HVPSU6.dll**. It is located in the directory "Program" of the enclosed software package. The 64-bit version is located in the subdirectory "x64". The software interface is a stand-alone software package, it does not require any additional library or driver, except the virtual port drivers (see section "Driver Installation"). Some library versions may also require the visual C++ runtime library, they can be freely downloaded from the Microsoft's homepage.

The user functions in the dynamic link library **COM-HVPSU6.dll** can be called from any popular programming language. For the details, please consult the user manual of your compiler. The definition of the interface functions is located in the C-header file **COM-HVPSU6.h**, it is independent on the bit version.

The sample program at the end of this section is written for the Borland C++ compiler. However, you should also be able to compile it with other C++ compilers without needing to modify it.

Functionality of the Software Interface

The software interface controls one communication channel for the data transfer from or to the device (see function **COM_HVPSU6_Open**). The communication channel must be opened before starting the communication. The open procedure configures the used virtual serial port and clears the port buffers.

The communication channel should be closed at the program end (see function **COM_HVPSU6_Close**). If this fails to happen, the software interface does it automatically for you when the dynamic link library **COM-HVPSU6.dll** is unloaded from the system memory.

The return value of most interface functions is a signed 32-bit number (**int**) describing the success of the particular operation. The last return value can be reloaded by the function **COM_HVPSU6_State**. Table Tab. 8 summarizes the possible return values together with the error messages, which can also be obtained by the function **COM_HVPSU6_ErrorMessage**. The error codes in Tab. 8 correspond to the codes **COM_HVPSU6_ERR_xxx** in the C-header file **COM-HVPSU6.h**. If a failure in data transfer has occurred, you can

discover the reason by calling the functions **`COM_HVPSU6_IO_State`** and **`COM_HVPSU6_IO_ErrorMessage`**. The first one returns the last I/O error, the second one the corresponding error message (see Tab. 9.).

If you have troubles understanding the errors or establishing the communication, please contact the manufacturer of the device.

Error Codes

Tab. 8. Return values of the interface functions

Return value	Error message	Description
0	No error	The data transfer finished successfully.
-2	Error opening the port	The port could not be opened. For the possible reasons, see Tab. 9.
-3	Error closing the port	The port could not be closed. For the possible reasons, see Tab. 9.
-4	Error purging the port	The port buffers could not be cleared.
-5	Error setting the port control lines	The port control lines could not be set.
-6	Error reading the port status lines	The port status lines could not be read.
-7	Error sending command	The data transfer to the device failed. For the possible reasons, see Tab. 9.
-8	Error sending data	
-9	Error sending termination character	
-10	Error receiving command	The data transfer from the device failed. For the possible reasons, see Tab. 9.
-11	Error receiving data	
-12	Error receiving termination character	
-13	Wrong command received	The device sent an unexpected response.
-14	Wrong argument received	
-15	Wrong argument passed to the function	One of the arguments passed to the function was out of the allowable range.

Return value	Error message	Description
-100	Device not connected	The port status lines indicate that the device is not connected.
-101	Device not ready	The port status lines indicate that the device is not ready. The communication with the device is possible only if it does not execute any process. Terminate all dialog boxes and menus at the device and retry the operation.
-102	Device state could not be set to not ready	The device did not react properly. Try to reset the communication or restart the device by powering it off and on.
-400	Error opening the file for debugging output	The file for debugging output cannot be opened for writing. Check if you have permissions to perform this action or if the file exists and is opened by another application.
-401	Error closing the file for debugging output	The file for debugging output cannot be closed. Check if the access to the file is still possible.

Tab. 9. I/O errors

Return value	Error message	Description
0	No error	The data transfer finished successfully.
1	Port has not been opened yet	You attempted to use the communication channel before having opened it.
2	Cannot open the port	The specified port could not be opened. Either the port does not exist or it is being currently used by another program.
3	Cannot get the state of the port	The system could not get the state of the port.
4	Cannot set the state of the port	The system could not set the state of the port.
5	Cannot set the timeouts for the port	The system could not set the timeouts for the port.
6	Cannot clear the port	The system could not clear the port buffers.
7	Error reading data from the port	The system could not read data from the port. Most probably, no data is available because the device is either disconnected or does not respond.
8	Error writing data to the port	The system could not write data to the port.
9	Wrong data amount written to the port	The system could not write the proper data amount to the port.
10	Error setting the control lines of the port	The system could not set the state of the port control lines.
11	Error reading the status lines of the port	The system could not get the state of the port status lines.
12	Device is busy	The system could not access the device since menus or dialog boxes are active.

Communication Control

Function COM_HVPSU6_Open

```
int COM_HVPSU6_Open (BYTE COMNumber);
```

Opens the communication channel and returns an error code according to Tab. 8.

The function opens the channel and attaches it to the serial port with the number **COMNumber**. The serial port number **COMNumber** must point to a valid serial port to that the controller is attached. Note that this number is the number of the COM port, i.e. **COMNumber = 1** points to the port COM1. The function accepts all numbers that are supported by the operating system, thus any port between COM1 and COM255 can be used for the communication.

You must call this function prior to any other communication function. If the function returns an error, no data communication is possible.

Function COM_HVPSU6_Close

```
int COM_HVPSU6_Close();
```

Closes the specified communication channel and returns an error code according to Tab. 8.

You can use the function to free the port for another application.

If an application that has exclusively used the software interface **COM-HVPSU6.dll** finishes, the opened communication channel closes automatically. Thus, the programmer does not need to call the function **COM_HVPSU6_Close** explicitly.

Function COM_HVPSU6_Purge

```
int COM_HVPSU6_Purge();
```

Clears the port data buffers and returns an error code according to Tab. 8.

The function can be used to repair a disturbed communication. In case of a crash of a user program, this function should be called to erase data incorrectly received from the device.

The function **COM_HVPSU6_Purge** is automatically called by the function **COM_HVPSU6_Open**.

Function COM_HVPSU6_Buffer_State

```
int COM_HVPSU6_Buffer_State (BOOL & Empty);
```

Gets the state of the device's input data buffer in the variable **Empty**. The return value is an error code according to Tab. 8.

When a large data amount should be transferred to the device, this function can be used to ensure that the input data buffer contains enough free space. If the return value of the variable **Empty** is false, the input buffer is not empty and there is no guarantee that the device will be able to receive the data. This situation can occur if the device has just received a large amount of data and its processing has not finished yet. In such case, the call to the function **COM_HVPSU6_Buffer_State** should be repeated after several milliseconds until the return value becomes **true**.

Function COM_HVPSU6_Device_Purge

```
int COM_HVPSU6_Device_Purge (BOOL & Empty);
```

Clears the device's output data buffer and gets the state of the device's input data buffer in the variable **Empty** like the function **COM_HVPSU6_Buffer_State**. The return value is an error code according to Tab. 8.

The function can be used to repair a disturbed communication. If the device does not respond properly, the function **COM_HVPSU6_Device_Purge** should be called repeatedly until it returns the value **true** in the variable **Empty**.

Device control

Function COM_HVPSU6_GetVoltage

```
int COM_HVPSU6_GetVoltage
  (unsigned int Channel, double & Voltage);
```

Gets the output voltage of the specified channel in the variable **Voltage**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the defines **COM_HVPSU6_CH_Vx**.

The return value in the variable **Voltage** is the set output voltage in Volts.

For more details, see section "Voltage Control".

Function COM_HVPSU6_SetVoltage

```
int COM_HVPSU6_SetVoltage
  (unsigned int Channel, double & Voltage);
```

Sets the output voltage of a specified channel according to the variable **Voltage**, gets the set value back in the same variable, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltage**.

The return value in the variable **Voltage** is the set output voltage in Volts. Note that the voltage is output only if the device is enabled.

After calling the function **COM_HVPSU6_SetVoltage**, you should check the return value in the variable **Voltage** to obtain the real value set by the device. If you, for instance, try to set an output voltage outside the specified range, the device truncates the value and the variable **Voltage** returns the modified value. Similarly, the value of the variable **Voltage** is adjusted to match the resolution of the respective module of the device.

For more details, see section "Voltage Control".

Function COM_HVPSU6_GetVoltageSlope

```
int COM_HVPSU6_GetVoltageSlope
  (unsigned int Channel,
  unsigned int & Slope);
```

Gets the slope of the output voltage of the specified channel in the variable **Slope**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltage**.

The return value in the variable **Slope** is the set output voltage slope in Volts per second.

For more details, see section "Voltage Slope Control".

Function COM_HVPSU6_SetVoltageSlope

```
int COM_HVPSU6_SetVoltageSlope
  (unsigned int Channel,
  unsigned int Slope);
```

Sets the slope of the output voltage of a specified channel according to the variable **Slope**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltage**.

The value in the variable **Slope** is the output voltage slope in Volts per second.

After calling the function **COM_HVPSU6_SetVoltageSlope**, you should call the function **COM_HVPSU6_GetVoltageSlope** and check its return value in the variable **Slope** to obtain the real value set by the device. If you, for instance, try to set an output voltage slope outside the specified range, the device truncates the value and the variable **Slope** returns the modified value.

For more details, see section "Voltage Slope Control".

Function COM_HVPSU6_GetVoltageLimit

```
int COM_HVPSU6_GetVoltageLimit
  (unsigned int Channel,
  unsigned int & Limit);
```

Gets the limit of the output voltage of the specified channel in the variable **Limit**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the defines **COM_HVPSU6_CH_VXX_LIMIT**.

The return value in the variable **Limit** is the output voltage limit in Volts.

For more details, see section "Voltage Limit Control".

Function COM_HVPSU6_SetVoltageLimit

```
int COM_HVPSU6_SetVoltageLimit
  (unsigned int Channel,
  unsigned int Limit);
```

Sets the set limit of the output voltage of a specified channel to the variable **Limit**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltageLimit**.

The value in the variable **Limit** is the output voltage limit in Volts.

After calling the function **COM_HVPSU6_SetVoltageLimit**, you should call the function **COM_HVPSU6_GetVoltageLimit** and check its return value in the variable **Limit** to obtain the real value set by the device. If you, for instance, try to set an output voltage limit outside the specified range, the device truncates the value and the variable **Limit** returns the modified value.

For more details, see section "Voltage Limit Control".

Function COM_HVPSU6_GetAmplitude

```
int COM_HVPSU6_GetAmplitude
  (unsigned int Channel,
  double & Amplitude);
```

Gets the amplitude of the output voltage of the specified channel in the variable **Amplitude**, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltage**.

The return value in the variable **Amplitude** is the set amplitude of the output voltage in V.

For more details, see section "Amplitude Control".

Function COM_HVPSU6_SetAmplitude

```
int COM_HVPSU6_SetAmplitude
  (unsigned int Channel,
  double & Amplitude);
```

Sets the amplitude of the output voltage of a specified channel according to the variable **Amplitude**, gets the set value back in the same variable, and returns an error code according to Tab. 8.

For the variable **Channel**, see the description of the function **COM_HVPSU6_GetVoltage**.

The return value in the variable **Amplitude** is the set amplitude of the output voltage in Volts.

After calling the function **COM_HVPSU6_SetAmplitude**, check the return value in the variable **Amplitude** to obtain the actual value set by the device. If, for instance, you try to set an amplitude that would lead to an output voltage outside the specified range, the device truncates the value and the variable **Amplitude** returns the modified value. Similarly, the value of the variable **Amplitude** is adjusted to match the resolution of the respective module of the device.

For more details, see section "Amplitude Control".

Function COM_HVPSU6_GetDeviceState

```
int COM_HVPSU6_GetDeviceState
  (unsigned int & DeviceState);
```

Gets the device state in the variable **DeviceState**, and returns an error code according to Tab. 8.

The return value in the variable **DeviceState** is a bit combination of the defines **COM_HVPSU6_LED_XXX**, **COM_HVPSU6_ENB_XXX**, and **COM_HVPSU6_ILOCK_XXX**.

For more details, see section "Voltage Control".

Function COM_HVPSU6_SetDeviceState

```
int COM_HVPSU6_SetDeviceState (bool HVEnable,
  unsigned int & DeviceState);
```

Enables or disables the device according to the variable **HVEnable**, gets the device state in the variable **DeviceState,** and returns an error code according to Tab. 8.

For the variable **DeviceState**, see the description of the function **COM_HVPSU6_GetDeviceState**.

If the device cannot be enabled - for instance, due to an opened interlock loop - the bit **COM_HVPSU6_ENB_PSU** in the variable **DeviceState** remains reset.

For more details, see section "Voltage Control".

Lock-in Control

Function COM_HVPSU6_GetLockinTiming

```
int COM_HVPSU6_GetLockinTiming
  (unsigned int Item, double & Timing);
```

Gets the lock-in timing of the item given by the variable **Item** in the variable **Timing**, and returns an error code according to Tab. 8.

For the variable **Item**, see the defines **COM_HVPSU6_TM_x**.

The return value in the variable **Timing** is the respective timing value in seconds. Note that the times are measured with a precision of 1/1024 s which results in a precision of about 1 ms.

For more details, see section "Lock-in Control".

Function COM_HVPSU6_SetLockinTiming

```
int COM_HVPSU6_SetLockinTiming
  (unsigned int Item, double & Timing);
```

Sets the lock-in timing of the item given by the variable **Item** to the variable **Timing**, and returns an error code according to Tab. 8.

For the variable **Item**, see the description of the function **COM_HVPSU6_GetLockinTiming**.

After calling the function **COM_HVPSU6_SetLockinTiming**, check the return value in the variable **Timing** to obtain the actual value set by the device. If, for instance, you try to set a timing value outside the specified range, the device truncates the value and the variable **Timing** returns the modified value. Similarly, the value of the variable **Timing** is adjusted to match the temporal resolution of 1/1024 s.

For more details, see section "Lock-in Control".

Function COM_HVPSU6_GetTargetFrames

```
int COM_HVPSU6_GetTargetFrames
  (unsigned int & TargetFrames);
```

Gets the number of target frames of a measurement in the variable **TargetFrames**, and returns an error code according to Tab. 8.

The return value in the variable **TargetFrames** is limit of measured frames. The measurement ends when the number of acquired frames reaches the value **TargetFrames**. For more details, see the value "Frames set" in the section "Measurement Control".

Function COM_HVPSU6_SetTargetFrames

```
int COM_HVPSU6_SetTargetFrames
  (unsigned int & TargetFrames);
```

Sets the number of target frames of a measurement to the variable **TargetFrames**, and returns an error code according to Tab. 8.

After calling the function **COM_HVPSU6_SetTargetFrames**, check the return value in the variable **TargetFrames** to obtain the actual value set by the device. It can be truncated if a value outside the specified range is passed to the function.

For more details, see the function **COM_HVPSU6_GetTargetFrames** and section "Measurement Control".

Function COM_HVPSU6_GetMeasState

```
int COM_HVPSU6_GetMeasState
  (unsigned int & MeasState,
  unsigned int & MeasStep, BOOL & MeasPhase,
  unsigned int & Frames);
```

Gets the measurement state in the variables **MeasState**, **MeasStep**, **MeasPhase**, and **Frames**, and returns an error code according to Tab. 8. The function is intended to poll the state of a running measurement.

The return value in the variable **MeasState** is the measurement state. For the allowed values, see the defines **COM_HVPSU6_MS_x**. For more details, see the value "State" in the section "Measurement Control".

The return value in the variable **MeasStep** is the measurement step, i.e. a part of a measurement frame. For the allowed values, see the defines **COM_HVPSU6_MP_x**. For more details, see the value "Step" in the section "Measurement Control".

The return value in the variable **MeasPhase** is the measurement phase, it is **false** during the first half of a measurement frame where the output voltages are set to the values in the dialog box "Voltage Control" (see Fig. 4). This is indicated by the output value "Vn" in the dialog box "Measurement Control" (see Fig. 10). The variable **MeasPhase** is **true** during the second half of a measurement frame where the output voltages are set to the values defined by adding up the voltages in the dialog box "Voltage Control" (see Fig. 4) and the amplitude values in the dialog box "Amplitude Control" (see Fig. 8). This is indicated by the output value "Vn+δVn" in the dialog box "Measurement Control" (see Fig. 10). For more details, see the value "Output" in the section "Measurement Control".

The return value in the variable **Frames** is the number of frames acquired so far. For more details, see the value "acquired" in the section "Measurement Control" as well as the functions **COM_HVPSU6_GetTargetFrames** and **COM_HVPSU6_SetTargetFrames**.

Function COM_HVPSU6_SetMeasState

```
int COM_HVPSU6_SetMeasState
  (unsigned int MeasCommand);
```

Changes the measurement state according the variable **MeasCommand**, and returns an error code according to Tab. 8. The function is intended to start or stop a measurement.

For the allowed values of the variable **MeasCommand**, see the defines **COM_HVPSU6_MC_x**.

After calling the function **COM_HVPSU6_SetMeasState**, check the device state by calling the function **COM_HVPSU6_GetMeasState**. Its return value in the variable **MeasState** can be evaluated to identify whether the issued command has produced the desired effect or not. The function **COM_HVPSU6_GetMeasState** can also be used to poll the state of a running measurement and identify when it is complete.

For more details, see section "Measurement Control".

Composite Functions

These functions combine several discrete function calls. The functionality is equal to the partial function calls, but the execution time is shorter because the latency time of each particular call is eliminated. The typical latency time of the used USB interface is around 15 ms, this enables about 60 function calls per second. By combining the function calls into one composite function, the latency is the same as for a single function call, even when several individual commands are issued. Utilizing the composite functions instead of frequent function calls can substantially increase the data throughput and shorten the response time.

Function COM_HVPSU6_SetVoltages

```
int COM_HVPSU6_SetVoltages
  (double Voltage [COM_HVPSU6_LENS_NUM]);
```

Sets the output voltages V1-V4 according to the variable array **Voltage**, gets the set values back in the same variable array, and returns an error code according to Tab. 8.

The functionality is equivalent to four subsequent calls to the function **COM_HVPSU6_SetVoltage**.

Function COM_HVPSU6_SetAmplitudes

```
int COM_HVPSU6_SetAmplitudes
  (double Amplitude [COM_HVPSU6_LENS_NUM]);
```

Sets the amplitude of the output voltages V1-V4 according to the variable array **Amplitude**, gets the set values back in the same variable array, and returns an error code according to Tab. 8.

The functionality is equivalent to four subsequent calls to the function **COM_HVPSU6_SetAmplitude**.

Function COM_HVPSU6_SetLockinTimings

```
int COM_HVPSU6_SetLockinTimings
  (double Timing [COM_HVPSU6_TM_NUM]);
```

Sets the lock-in timings to the variable array **Timing**, and returns an error code according to Tab. 8.

The functionality is equivalent to four subsequent calls to the function **COM_HVPSU6_SetLockinTiming**.

Monitoring

Function COM_HVPSU6_GetMeasuredVoltages

```
int COM_HVPSU6_GetMeasuredVoltages
  (double Voltage [COM_HVPSU6_VOLT_NUM]);
```

Gets the measured output voltages in the array variable **Voltage**, and returns an error code according to Tab. 8.

The return value in the variable **Voltage** are the measured output voltages in Volts. Note that the voltages are measured with a 16-bit precision so that the measured voltage values may slightly differ from the values obtained by the functions **COM_HVPSU6_GetVoltage** and **COM_HVPSU6_SetVoltage**.

For more details, see section "Voltage Monitor".

Function COM_HVPSU6_Housekeeping

```
int COM_HVPSU6_Housekeeping
  (double & Volt12V, double & Volt5V0,
  double & Volt3V3, double & VoltAgnd,
  double & TempCPU);
```

Gets the housekeeping data and returns an error code according to Tab. 8.

The return values in the variables **Volt12V**, **Volt5V0**, and **Volt3V3** are the supply voltages in Volts. The return value in the variable **VoltAgnd** is the voltage of the analog ground in Volts. The return value in the variable **TempCPU** is the microcontroller temperature in °C.

For more details, see section "Supply Monitor".

Function COM_HVPSU6_HousekeepingSMPS

```
int COM_HVPSU6_HousekeepingSMPS
  (double & Volt12V, double & Volt5V0,
  double & Volt3V3, double & TempCPU);
```

Gets the housekeeping data of the module HV-SMPS2 and returns an error code according to Tab. 8.

For the return values, see the function **COM_HVPSU6_Housekeeping**.

For more details, see section "Supply Monitor".

Function COM_HVPSU6_GetMeasuredVoltages

```
int COM_HVPSU6_GetMeasuredTemperatures
  (double
  Temperature [COM_HVPSU6_TEMP_NUM]);
```

Gets the measured temperatures in the array variable **Temperature**, and returns an error code according to Tab. 8.

The return value in the variable **Temperature** are the measured temperatures in °C. For the assignment to the sensors, see the defines **COM_HVPSU6_TEMP_XXX**.

For more details, see section "Temperature Monitor".

Error Handling

Function COM_HVPSU6_State

```
int COM_HVPSU6_State();
```

Returns the state of the software interface according to Tab. 8.

This function can be used to obtain the last return value of an interface function.

The function does not have any influence on the communication and can be called at any time.

Function COM_HVPSU6_ErrorMessage

```
const char * COM_HVPSU6_ErrorMessage();
```

Returns the error message corresponding to the state of the software interface. The return value is a pointer to a null-terminated character string according to Tab. 8.

The function does not have any influence on the communication and can be called at any time.

Function COM_HVPSU6_IO_State

```
int COM_HVPSU6_IO_State();
```

Returns the interface state of the serial port according to Tab. 9.

The function can be used to obtain the result of the last I/O operation at the port.

The function does not have any influence on the communication and can be called at any time.

Function COM_HVPSU6_IO_ErrorMessage

```
const char * COM_HVPSU6_IO_ErrorMessage();
```

Returns the error message corresponding to the interface state of the serial port. The return value is a pointer to a null-terminated character string according to Tab. 9.

The function does not have any influence on the communication and can be called at any time.

Various Functions

Function COM_HVPSU6_SW_Version

```
WORD COM_HVPSU6_SW_Version();
```

Returns the version of the software interface (the dynamic link library **COM-HVPSU6.dll**).

The function should be used to check whether a software interface with the correct version is being used. The function should be called prior to any other function of the software interface.

The return value is an unsigned 16-bit integer (**WORD**). The higher byte contains the main version number, the lower byte the subversion, i.e. the version order within the main version.

Function COM_HVPSU6_GetCPUdata

```
int COM_HVPSU6_GetCPUdata (double & Load,
  double & Frequency);
```

Gets the controller CPU load and operation frequency in the variables **Load** and **Frequency**. The function return value is an error code according to Tab. 8.

The variable **Load** is the CPU load ranging from 0 (no load, i.e. 0%) to 1 (maximum load, i.e. 100%). The variable **Frequency** is the frequency in Hz of the clock controlling the peripheral devices of the microcontroller.

Function COM_HVPSU6_GetSMPSCPUdata

```
int COM_HVPSU6_GetSMPSCPUdata (double & Load,
  double & Frequency);
```

Gets the CPU load and operation frequency of the microcontroller in the module HV-SMPS2 in the variables **Load** and **Frequency**. The function return value is an error code according to Tab. 8.

For the return values, see the function **COM_HVPSU6_GetCPUdata**.

For more details, see section "Clock Monitor".

Function COM_HVPSU6_GetUptime

```
int COM_HVPSU6_GetUptime  (DWORD & Seconds,
  WORD & Milliseconds, DWORD & Optime);
```

Gets the device uptime and operation time in the variables **Seconds**, **Milliseconds**, and **Optime**. The function return value is an error code according to Tab. 8.

The device uptime is the time elapsed from the last (re)start of the device. The value of the uptime can be used, for instance, to see whether the device restarted unexpectedly.

The device operation time is the time period during which the device was activated since last (re)start.

For more details, see section "Statistics".

Function COM_HVPSU6_GetUptime

```
int COM_HVPSU6_GetTotalTime (DWORD & Uptime,
  DWORD & Optime);
```

Gets the total device uptime and total operation time in the variables **Uptime** and **Optime**. The function return value is an error code according to Tab. 8.

The total device uptime is the total time period during which the device was powered on. The total device operation time is the time period during which the device was activated.

For more details, see section "Statistics".

Function COM_HVPSU6_HW_Version

```
int COM_HVPSU6_HW_Version (WORD & Version);
```

Gets the device's hardware version in the variable **Version**. The function return value is an error code according to Tab. 8.

The return value in the variable **Version** should be used to check whether the hardware has an appropriate version.

The return value is similar to the return value of the function **COM_HVPSU6_SW_Version**. It is an unsigned 16-bit integer (**WORD**) containing the main version and the subversion numbers.

For more details, see section "System Information".

Function COM_HVPSU6_FW_Version

```
int COM_HVPSU6_FW_Version (WORD & Version);
```

Gets the device's firmware version in the variable **Version**. The function return value is an error code according to Tab. 8.

The return value in the variable **Version** should be used to check whether the software interface is in an appropriate version.

The return value is similar to the return value of the function **COM_HVPSU6_SW_Version**. It is an unsigned 16-bit integer (**WORD**) containing the main version and the subversion numbers.

For more details, see section "System Information".

Function COM_HVPSU6_FW_Date

```
int COM_HVPSU6_FW_Date (char * DateString);
```

Gets the device's firmware date in the variable **DateString**. The function return value is an error code according to Tab. 8.

The return value in the variable **DateString** is a null-terminated character string with the firmware compilation date. The buffer passed to the function must be created before the function call, it must be at least 16 bytes long.

For more details, see section "System Information".

Function COM_HVPSU6_Product_No

```
int COM_HVPSU6_Product_No (DWORD & Number);
```

Gets the device's product number in the variable **Number**. The function return value is an error code according to Tab. 8.

The return value in the variable **Number** is the product number as an unsigned 32-bit integer (**DWORD**).

For more details, see section "System Information".

Function COM_HVPSU6_Product_ID

```
int COM_HVPSU6_Product_ID
  (char * Identification);
```

Gets the device's product identification in the variable **Identification**. The function return value is an error code according to Tab. 8.

The return value in the variable **Identification** is a null-terminated character string with the product identification. The buffer passed to the function must be created before the function call; it should be 81 bytes long.

For more details, see section "System Information".

Sample Communication Program

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include "COM-HVPSU6.h"

// handle a possible error:
static int Error (int code)
   {
   printf ("%s\n", COM_HVPSU6_ErrorMessage());
   return code;
   }

static char * MakeLED (bool active)
   {
   return active ? "On" : "Off";
   }

int main (int argc, char * argv[])
   {
   if (argc != 2)
      {
      printf ("The number of the COM port must be specified!\n");
      return -1;
      }
   // check the DLL version:
   const WORD DLL_Version = COM_HVPSU6_SW_Version();
   if ((DLL_Version & 0xFF00) != 0x0100)
      {
      printf ("Wrong DLL Version: expected 1.x, found %d.%02d\n",
         DLL_Version / 0x100, DLL_Version & 0xFF);
      return 1;
      }

   // open the port:
   WORD COMNumber;
   sscanf (argv[1], "%hu", &COMNumber);
   if (COM_HVPSU6_Open (COMNumber))
      return Error (2); // handle a possible error

   // get the product number:
   DWORD Number;
   if (COM_HVPSU6_Product_No (Number))
      return Error (10); // handle a possible error
   printf ("Product number: %u\n", Number);
```

```
// get the device state:
unsigned int DeviceState;
if (COM_HVPSU6_GetDeviceState (DeviceState))
   return Error (11); // handle a possible error
printf ("Device state: %04Xh\n", DeviceState);
printf ("LED Failure: %s\n",
   MakeLED (DeviceState & COM_HVPSU6_LED_FAIL));
printf ("LED Activated: %s\n",
   MakeLED (DeviceState & COM_HVPSU6_LED_ACT));
printf ("LED Interlock: %s\n",
   MakeLED (DeviceState & COM_HVPSU6_LED_ILOCK));

// activate the device:
if (DeviceState & COM_HVPSU6_ENB_PSU)
   printf ("Device already activated\n");
else
   {
   if (COM_HVPSU6_SetDeviceState (true, DeviceState))
      return Error (12); // handle a possible error
   if (DeviceState & COM_HVPSU6_ENB_PSU)
      printf ("Device activated\n");
   else
      printf ("Device could not be activated!\n");
   }

// set the output voltage V1
double Voltage = -10.23; // V1 = -10.23V
if (COM_HVPSU6_SetVoltage (COM_HVPSU6_CH_V1, Voltage))
   return Error (13); // handle a possible error
printf ("V1 set to %.3fV\n", Voltage);

// get the measured high-voltage values
printf ("Reading measured voltages...\n");
printf ("Press any key to stop\n");
while (true) {
   double Voltage [COM_HVPSU6_VOLT_NUM];
   if (COM_HVPSU6_GetMeasuredVoltages (Voltage))
      return Error (14); // handle a possible error
   printf ("Measured V1: %.3fV\n", Voltage[COM_HVPSU6_CH_V1]);
   if (kbhit()) break;
   Sleep (500);
   }
getch();

// close the port:
if (COM_HVPSU6_Close())
   return Error (2); // handle a possible error

//finish the program:
return 0;
}
```