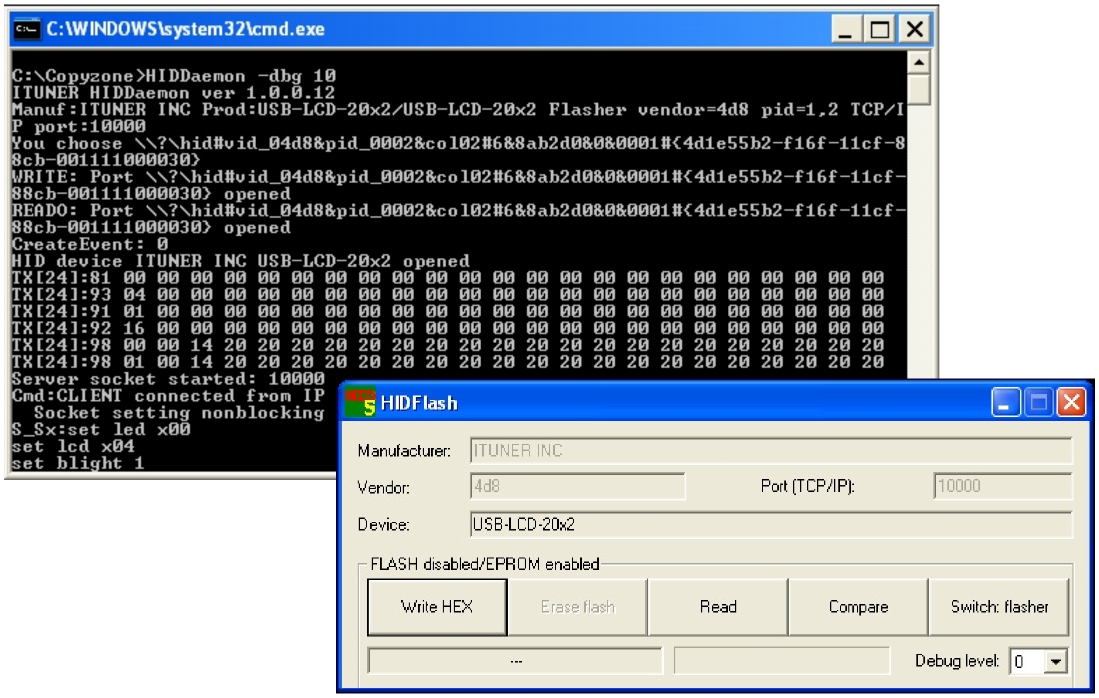
USB-LCD-20x2 driver software specification



C:\WINDOWS\system32\cmd.exe

```
C:\Copyzone>HIDDaemon -dbg 10
ITUNER HIDDaemon ver 1.0.0.12
Manuf:ITUNER INC Prod:USB-LCD-20x2/USB-LCD-20x2 Flasher vendor=4d8 pid=1,2 TCP/I
P port:10000
You choose \\?\hid#vid_04d8&pid_0002&col02#6&8ab2d0&0&0001#{4d1e55b2-f16f-11cf-8
8cb-001111000030}
WRITE: Port \\?\hid#vid_04d8&pid_0002&col02#6&8ab2d0&0&0001#{4d1e55b2-f16f-11cf-
88cb-001111000030} opened
READO: Port \\?\hid#vid_04d8&pid_0002&col02#6&8ab2d0&0&0001#{4d1e55b2-f16f-11cf-
88cb-001111000030} opened
CreateEvent: 0
HID device ITUNER INC USB-LCD-20x2 opened
TX[24]:81 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:93 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:91 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:92 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:98 00 00 14 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
TX[24]:98 01 00 14 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
Server socket started: 10000
Cmd:CLIENT connected from IP
  Socket setting nonblocking
S_Sx:set led x00
set lcd x04
set blight 1
```

HIDFlash

Manufacturer: ITUNER INC

Vendor: 4d8

Port (TCP/IP): 10000

Device: USB-LCD-20x2

FLASH disabled/EPROM enabled

Debug level: 0

● Introduction

The USB-LCD-20x2 module was designed to be a standalone user display and input device with USB connection usable in different applications where displaying texts or reading some buttons was needed. This device could be connected to any personal or embedded computer having at least one USB connector available.

The USB-LCD-20x2 device can display texts, can receive button pressing or infrared remote controller emitted signals and can start or shut down a computer from simple button pressing or from remote control button pressing.

The LCD screen has ON/OFF controllable backlight source and variable contrast level.

You can also define 8 custom characters and save them to the modules Character Generator RAM (CG-RAM).

- **Important Note**: The Character Generator RAM is a volatile memory zone and after a general power off its content will loose!

You can define five splash screen contents too. The splash screen contents are text strings and they will be always displayed when the USB-LCD-20x2 device starts. The screen contents can be allowed or denied and shown for different intervals of time.

- **Important Note**: The Splash Screen text and settings are stored in the internal EEPROM, a non-volatile memory zone of the LCD device, therefore after a general power off the device will store its splash screen content.
-

Other important future of this device is the accessible program memory of the main controller. When the device starts in or changed to flasher mode of operation new or modified firmware versions could be downloaded into the USB-LCD-20x2device.

To allow all this possibilities the USB-LCD-20x2 module is shipped with installation CD containing three, basic software. Two of them will act as device driver and network server; the third software is a demo version GUI (Graphical User Interface) to demonstrate the USB-LCD-20x2 module’s hardware potentials.

- The first device driver, called USBLCDServer.exe is a text mode software version and allows different local commands as text strings displaying, LED driver outputs controlling, LCD backlight and contrast setting, digital inputs reading. The software allows also remote commands as well as locally introduced commands.
- The second device driver, called USBLCDFlasher.exe is GUI based software and it is used especially for upgrade the firmware version in the main controller. This program could act as network server as well as the USBLCDServer.exe.

Start and use USBLCDServer.exe

To start up testing you must follow a few easy steps as follows:

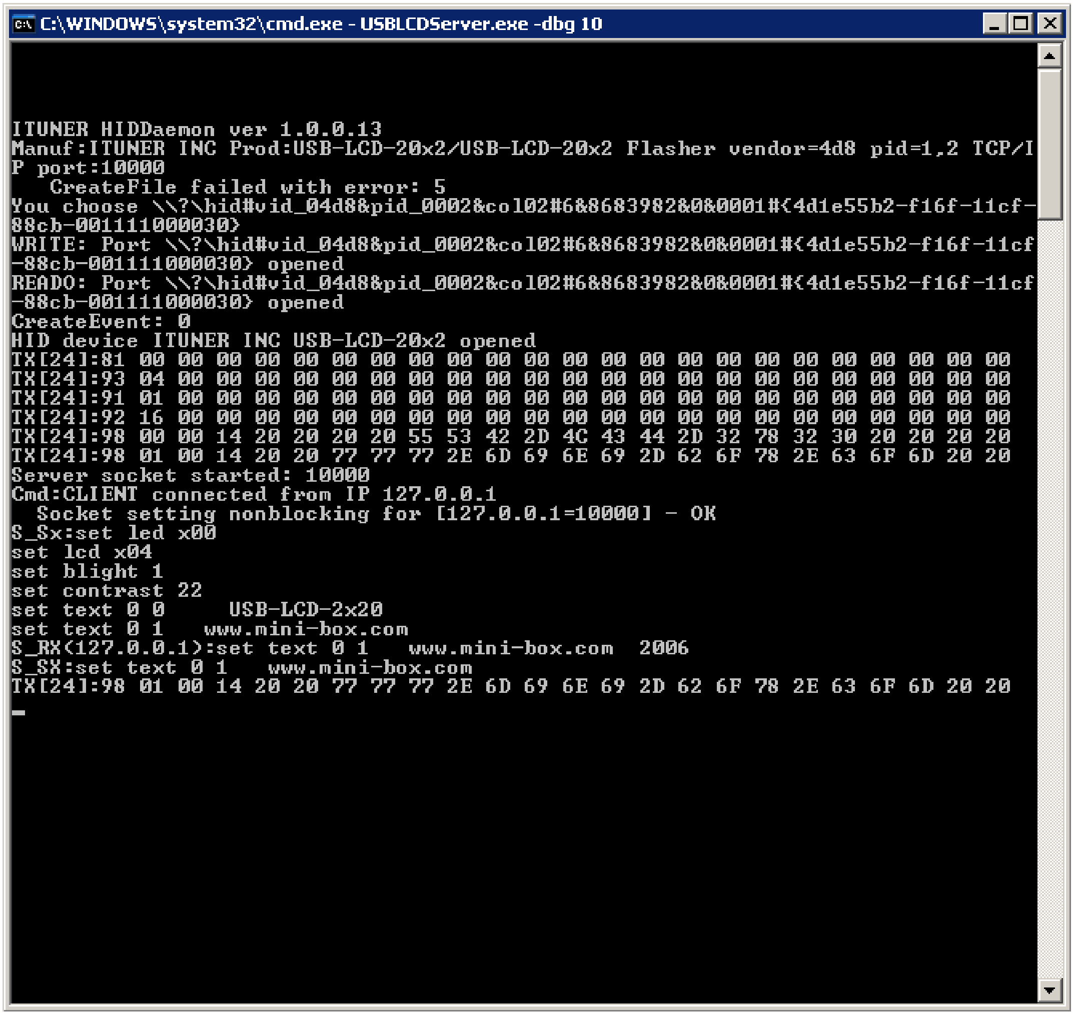
- Plug in the USB-LCD-20x2 device in a free USB connector or if it is already plugged turn on your computer
- Find and start the USBLCDServer.exe

At starting this software you may consider some parameters in command line, which can be tampering the execution mode.

You could start the USBLCDServer.exe with following parameters:

● **–manuf**	data type: string	manufacturer
● **–vendor**	data type: unsigned short	vendor ID
● **–keyb**	data type: string	device name in normal mode
● **–flash**	data type: string	device name in flasher mode
● **–dbg**	data type: unsigned short	debug level
● **–port**	data type: unsigned int	server socket port
● **–tout**	data type: unsigned int	timeout in ms when switching between Flasher and Keyboard mode

Once started the USBLCDServer.exe acts by default as network server, therefore no special setting are needed to allow this function. The user could set up the server socket port number optionally, if the default value: port 10000 is not available.



```
ITUNER HIDDaemon ver 1.0.0.13
Manuf:ITUNER INC Prod:USB-LCD-20x2/USB-LCD-20x2 Flasher vendor=4d8 pid=1,2 TCP/I
P port:10000
   CreateFile failed with error: 5
You choose \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf-
88cb-001111000030}
WRITE: Port \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf
-88cb-001111000030} opened
READO: Port \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf
-88cb-001111000030} opened
CreateEvent: 0
HID device ITUNER INC USB-LCD-20x2 opened
TX[24]:81 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:93 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:91 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:92 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:98 00 00 14 20 20 20 20 55 53 42 2D 4C 43 44 2D 32 78 32 30 20 20 20 20
TX[24]:98 01 00 14 20 20 77 77 77 2E 6D 69 6E 69 2D 62 6F 78 2E 63 6F 6D 20 20
Server socket started: 10000
Cmd:CLIENT connected from IP 127.0.0.1
  Socket setting nonblocking for [127.0.0.1=10000] - OK
S_Sx:set led x00
set lcd x04
set blight 1
set contrast 22
set text 0 0     USB-LCD-2x20
set text 0 1   www.mini-box.com
S_RX(127.0.0.1):set text 0 1   www.mini-box.com  2006
S_SX:set text 0 1   www.mini-box.com
TX[24]:98 01 00 14 20 20 77 77 77 2E 6D 69 6E 69 2D 62 6F 78 2E 63 6F 6D 20 20
_
```

● Implemented functions in USBLCDServer.exe

The USBLCDServer.exe implements a few commands, which can be introduced by user in the program command line. If the program will be accessed remotely all messages must be ended with null (string terminator null character) and they are received in the same way in the response strings.

There are three major types of messages, the **SET**, **GET** and the **NOTIFY** type messages. All this messages are presented below in the same format: command name, list of parameters, short description. The command lines and parameters are written in bold characters.

● SET message types

The SET messages will work only in **device mode** (not in flasher mode too). This means if the USB-LCD-20x2 device is in flasher mode this message will result in "**err flasher**" response.

All the messages without enough parameters, or wrong type of parameters, or inexistent messages are simply thrown away without any response or notification.

- **set led xFF**

  Description: FF is a hexadecimal number where every bit represents one LED from the device (imaginary consider the binary representation of the byte contents: 0 0 0 0 0 0 0 0, if the LED output 0 would be turn ON consider 0 0 0 0 0 0 0 1 which is simply x01 in the command line)

  Local or remote response: **set led xFF**

- **set blight N**

  Description: N = 1 or N = 0 LCD backlight switched ON/OFF

  Local or remote response: **set blight N**

- **set contrast N**

  Description: N = 0 to 64 defines the LCD contrast level

  Local or remote response: **set contrast N**

-

- **set clear**

Description: clear the LCD screen, all displayed texts will be deleted irremediable, the LCD character memory will be erased, the character location counter will be cleared to 0.
Local or remote response: **set clear**

- **set text x y "text"**

Description: x, y are the text coordinates, x represents the character location number (0..19), the y represents the display line number (0..1). Text can contain any displayable character (the " character at the start and end of each text string are necessary otherwise text might be truncated).
The device hardware permits special (or user defined) characters to displaying, i.e. \\, for slash character, \0, \1, \2,.., \15 to access and display the special characters (the special characters are mapped twice, number 0 to 7 are repeated from 8 to 15).

Local or remote response: **set text 0 y "text"** which means in the response the whole row content will be returned to avoid wrong calculation of character position in the client software.

- **set font no f0 f1 f2 f3 f4 f5 f6 f7**

Description: **no** parameter points the number of the special character to be stored in the LCD module CGRAM, can be defined in the 0..7. The f0..f7 noted number represents the 8 line of the character bitmap (each character is represented in a 5 column and 8 rows dot matrix.

  - **Important Notes:** for user definable character please refer to the attached documentation of the LCD module, especially read the chapter about CGRAM configuration and character representation.

Local or remote response: **none**

- **set relay msec**

Description: turns on or off the mini switch on the microcontroller mainboard. Timeout is in **milliseconds**, until switching off the device from the moment when it turns on. When a command is received it will be executed immediatelly and the mini switch turns on. If time value is 0x0000, it means instant "OFF", if the value is 0xFFFF, it means instant "ON" forewer (or until the next command received containing time value equal with 0x0000 or other value). Any value between the two extreme number set an n **msec** "ON" stage.

Local or remote response: **none**

- **set inte MIN|SEC|LEDs|20char|20char|MIN|SEC|...**

Description: one of the most complicated commands, used to write the preset timetable|text string into the main controllers internal EEPROM. **MIN|SEC|20char line|20char** line, repeated 5 times will represent the 5 preset screen content and their display timing.

Important Notes: MIN|SEC permits their values to be:

-1|-1	for infinite displaying of the screen content
0|0	for disabling the screen content (it will be simply skipped)

Important Notes: LEDs from LSBit to MSBit every bit controls a LED or LCD backlight:
Bit0: +/-/Up/Dn/Left/Right/OK LED group
Bit1: F1 key
Bit2: F2 key
Bit3: F3 key
Bit4: F4 key
Bit5: F5 key
Bit6: unused
Bit7: LCD backlight

- **set exte MIN|SEC|LEDs|20char|20char|MIN|SEC|...**

Description: one of the most complicated commands, used to write the preset timetable|text string into the external atached EEPROM. **MIN|SEC|20char line|20char** line, repeated 5 times will represent the 5 more preset screen content and their display timing.

Important Notes: MIN|SEC permits their values to be:

-1|-1	for infinite displaying of the screen content
0|0	for disabling the screen content (it will be simply skipped)

Important Notes: LEDs from LSBit to MSBit every bit controls a LED or LCD backlight:
Bit0: +/-/Up/Dn/Left/Right/OK LED group
Bit1: F1 key
Bit2: F2 key
Bit3: F3 key
Bit4: F4 key
Bit5: F5 key
Bit6: unused
Bit7: LCD backlight

● GET message types

The GET messages will work only in **device mode** (not in flasher mode too). This means if the USB-LCD-20x2 device is in flasher mode this message will result in "**err flasher**" response.

All the messages without enough parameters, or wrong type of parameters, or inexistent messages are simply thrown away without any response or notification.

- **get inte**

Description: loads the internal EEPROM's content into the "eprom_read.hex" named local file and from there the dates will be parsed in the same format as presented in the set inte message section.

Local or remote response: **get inte MIN|SEC|20char|20char|MIN|SEC|...** line, repeated 5 times will represents the 5 preset screen content and their display timing (please consider bottom lines on the page 5).

● NOTIFY message types

This type of message is always originated from the daemon software never from the client side. This message represents the input data streams from the USB-LCD-20x2 module. In this way any button pressing or any IR signal input triggering will be notified to the upper software level.

- **notify key xFF xFF**

Description: one of the USB-LCD-20x2 modules digital inputs was triggered (pulled to the GND) If there was a keyboard connected to the USB-LCD-20x2 module it means a key was pressed. Two inputs could be triggered in the same time. **xFF xFF** are the two, triggered inputs code in hexadecimal representation.

- **notify ir LEN xFF xFF ...**

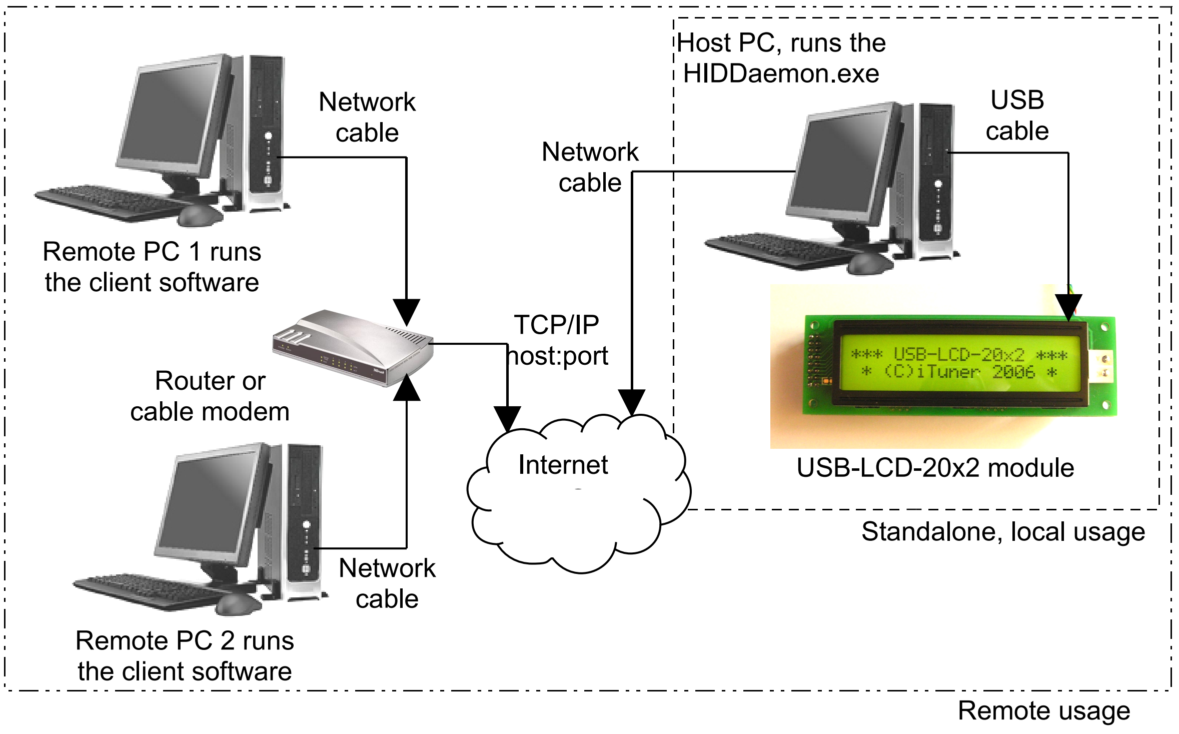
Description: the infrared input was triggered, some infrared signal activities were detected in the IR sensor viewing area, a possible message was received by the USB-LCD-20x2 module. **LEN** represents the length of the message body in tokens (numbers) - maximum 256 tokens could be allowed. **xFF xFF** ... LEN number of hexadecimal values representing the IR message as time intervals between two adjacent signal edges. In other words it means the captured IR signal was measured between two edges and the time interval in milliseconds will be converted in integer representation (xFF, xFF the lower byte, the upper byte). The time intervals longer than 65535ms will be ignored as non-contiguous IR signal flow.

● Establishing a remote connection with USBLCDServer.exe

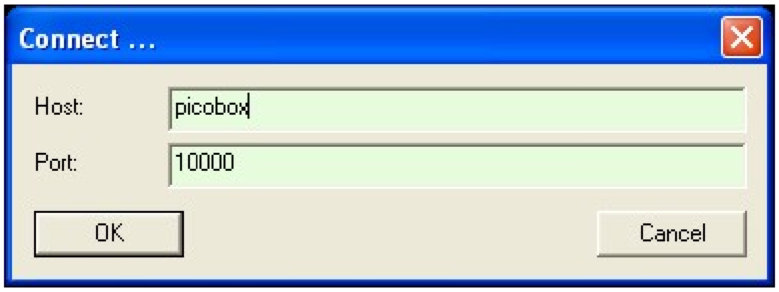
If the USB-LCD-20x2 module is not physically connected in the USB port of the user PCs a remote connection could be established over the local network or Internet connection. The basic requirements are: a powered PC with a connected USB-LCD-20x2 module and running USBLCDServer.exe software on the same PC.

The USBLCDServer.exe acts as a remote server and it will share a previously configured server socket port (default value 10000). From this moment the USBLCDClient.exe or a telnet type user interface could anytime connecting to host:port as defined. In fact once USBLCDServer.exe is running any connection to USB-LCD-20x2 goes trough it. There is no matter if a locally or remote connection was established before.

The following schematic shows one of the possible connection diagrams.



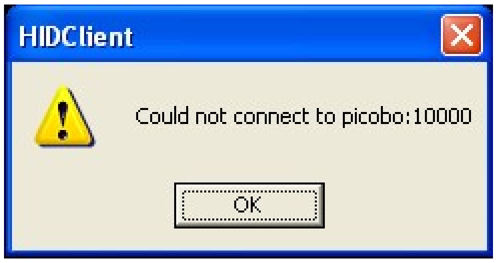
To build a same configuration connects an USB-LCD-20x2 module to a PCs USB port and start the USBLCDServer.exe on the same PC. This PC will be the Host node. Now you can run client software on this PC connecting to **localhost**, **port 10000** (or other available port, user configurable). If you would access the Host node remotely, from other PC, you must remember the Host name and port number before you start the client software. Once started the client software will resolve the connection between Host PC and Remote PC. In this way the network layer will be prepared and the USB-LCD-20x2 module becomes remotely usable.


Connect ...

Host: picobox

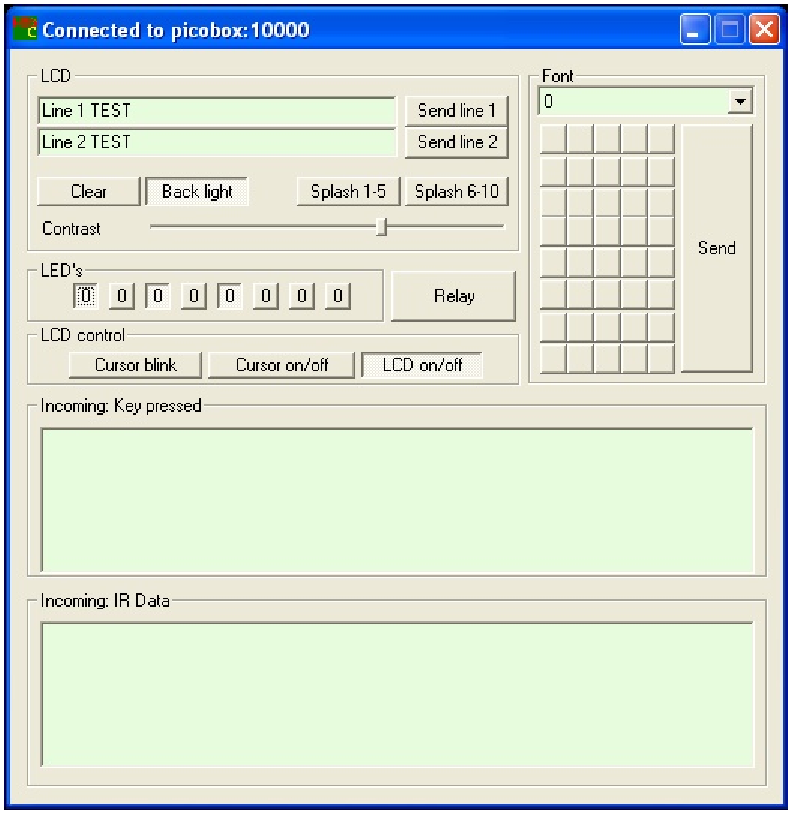
Port: 10000

As you can see in the picture a Host name was typed in the input field “picobox” and the Port number was left unchanged. By pressing OK the client software starts and will try to establish the remote connection. If the client software fails connection an error message will appear:


HIDClient

Could not connect to picobo:10000

If connection could be established the GUI starts to run:


Connected to picobox:10000

LCD

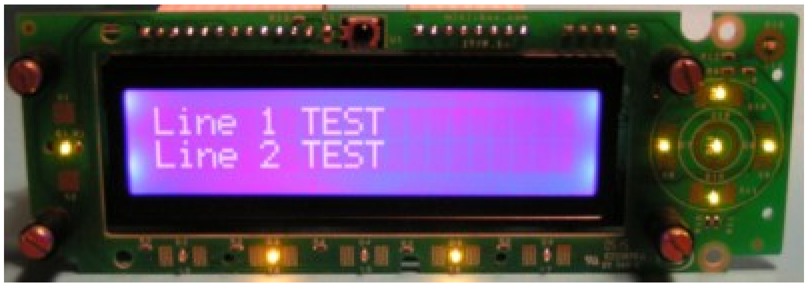
Line 1 TEST

Line 2 TEST

Font: 0

● Configuring and using the USB-LCD-20x2 module remotely

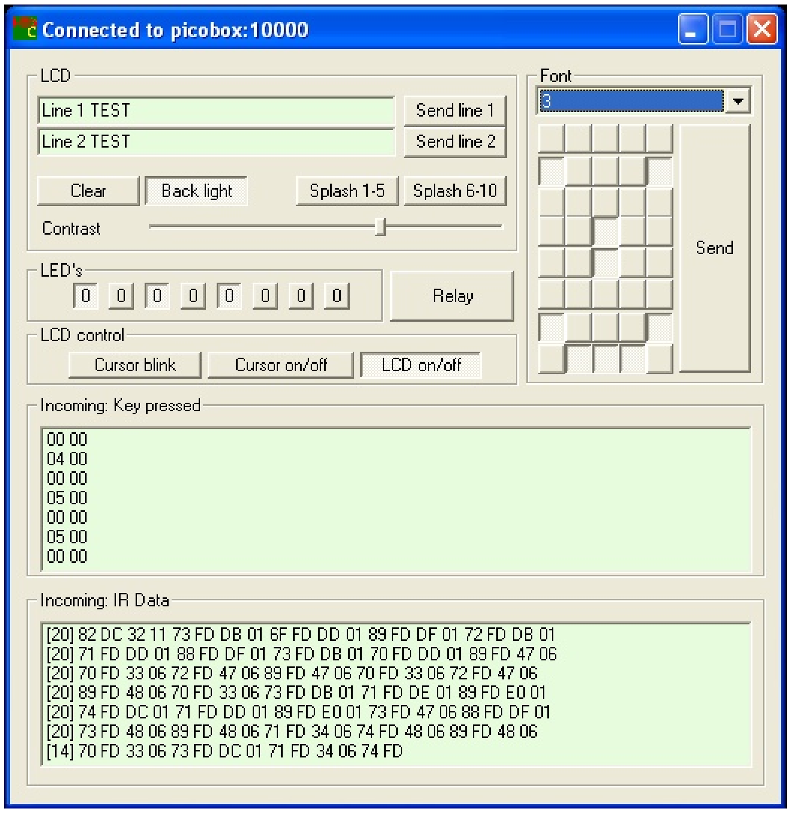
The first two input fields represent the actual LCD screen contents. You can type a 20 character long text (including spaces or special symbols) in each input fields and after by pressing the “**Send line 1**” or “**Send line 2**” button these text will be copied on the LCD screen, and will shown like this:



The “**Clear**” button deletes the whole LCD screen content. By pressing it all displayed text will disappear from the LCD screen.

The “**Backlight**” button serves to switch ON and OFF the LCD backlight. The immediate effect of pressing this button you will observe on the LCD screen. Please consider the backlight is a LED based solution so don’t expect to lights up your room. The backlight feature helps in dark or weak illuminated places.

The “**Contrast**” slide bar can be used to set up a proper contrast ratio for the LCD (depends on viewing angles).



Line 1 TEST

Line 2 TEST

Font: 3

LED's: 0 0 0 0 0 0 0 0

Incoming: Key pressed

```
00 00
04 00
00 00
05 00
00 00
05 00
00 00
```

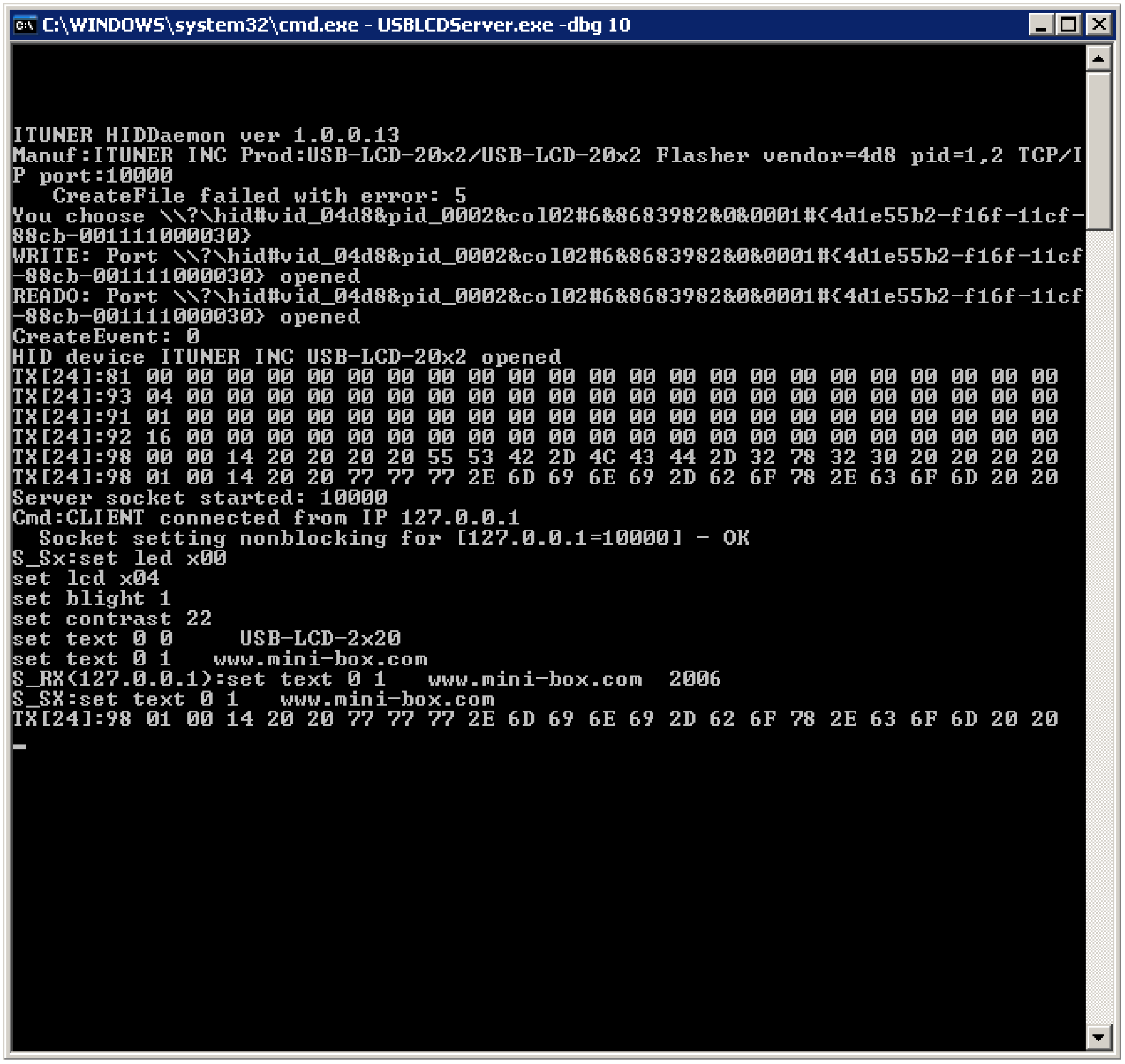
Incoming: IR Data

```
[20] 82 DC 32 11 73 FD DB 01 6F FD DD 01 89 FD DF 01 72 FD DB 01
[20] 71 FD DD 01 88 FD DF 01 73 FD DB 01 70 FD DD 01 89 FD 47 06
[20] 70 FD 33 06 72 FD 47 06 89 FD 47 06 70 FD 33 06 72 FD 47 06
[20] 89 FD 48 06 70 FD 33 06 73 FD DB 01 71 FD DE 01 89 FD E0 01
[20] 74 FD DC 01 71 FD DD 01 89 FD E0 01 73 FD 47 06 88 FD DF 01
[20] 73 FD 48 06 89 FD 48 06 71 FD 34 06 74 FD 48 06 89 FD 48 06
[14] 70 FD 33 06 73 FD DC 01 71 FD 34 06 74 FD
```

On the Host side the USBLCDServer.exe will notify you every incoming command and the given response message.

- **Important notes**: To see all of this command echoes on the Host side, don't forget to start the USBLCDServer.exe with the **–dbg** 10 command line option.

For example in case of "**Send line 1**" or "**Send line 2**" from USBLCDClient.exe you will observe in the USBLCDServer.exe window the following notifications:



```
ITUNER HIDDaemon ver 1.0.0.13
Manuf:ITUNER INC Prod:USB-LCD-20x2/USB-LCD-20x2 Flasher vendor=4d8 pid=1,2 TCP/I
P port:10000
   CreateFile failed with error: 5
You choose \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf-
88cb-001111000030}
WRITE: Port \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf
-88cb-001111000030} opened
READO: Port \\?\hid#vid_04d8&pid_0002&col02#6&8683982&0&0001#{4d1e55b2-f16f-11cf
-88cb-001111000030} opened
CreateEvent: 0
HID device ITUNER INC USB-LCD-20x2 opened
TX[24]:81 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:93 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:91 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:92 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
TX[24]:98 00 00 14 20 20 20 20 55 53 42 2D 4C 43 44 2D 32 78 32 30 20 20 20 20
TX[24]:98 01 00 14 20 20 77 77 77 2E 6D 69 6E 69 2D 62 6F 78 2E 63 6F 6D 20 20
Server socket started: 10000
Cmd:CLIENT connected from IP 127.0.0.1
  Socket setting nonblocking for [127.0.0.1=10000] - OK
S_Sx:set led x00
set lcd x04
set blight 1
set contrast 22
set text 0 0     USB-LCD-2x20
set text 0 1   www.mini-box.com
S_RX(127.0.0.1):set text 0 1   www.mini-box.com  2006
S_SX:set text 0 1   www.mini-box.com
TX[24]:98 01 00 14 20 20 77 77 77 2E 6D 69 6E 69 2D 62 6F 78 2E 63 6F 6D 20 20
_
```

S_RX(192.168.1.6):set text 0 0 "Line 1 TEST"
– remote command notification, put text message in the first row of the LCD, the remote PCs IP will shown first

S_SX:set text 0 0 "Line 1 TEST   "

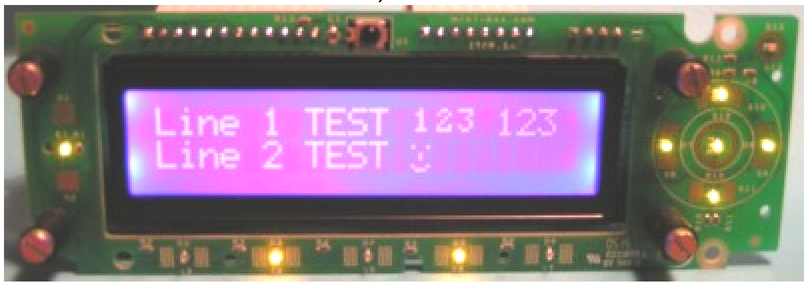
– local command execution echo

TX[24]:98 00 00 0B 4C 69 6E 65 20 31 20 54 45 53 54 00 00 00 00 00 00 00 00 00
– the effective USB level command packet, this series of hexadecimal numbers will be sent to the USB-LCD-20x2 device trough the USB connection

This content will be repeated in case of pressing "**Send line 2**" GUI button on the remote side.

In the "**LED's**" field are some buttons, which can turn ON and OFF the buttons backlights and if pressing them you will observe in the USBLCDServer.exe window the following notifications:

S_RX(192.168.1.6):set led x01
– remote command notification, turn on the LED number one (x01), the



remote PCs IP will shown first

S_SX:set led x01
– local command execution echo

TX[24]:81 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
– the effective USB level command packet, this series of hexadecimal numbers will be sent to the USB-LCD-20x2 device trough the USB connection

As result of command execution you will observe some of the LED diodes turns on. If there are no LEDs connected to the digital outputs of the USB-LCD-20x2, you could measuring a positive 5V voltage level on the corresponding digital output pin

- **Important notes:** please refer to the iTuner USB-LCD-20x2 module technical notes No.1. about hardware setup and pinouts of the LCD module.

● USB-LCD-20x2 low-level command messages description

The last chapter of the documentation contains the low-level command descriptions regarding the implemented functions in the USB-LCD-20x2 module. Using this command set, the user could be able to write a proprietary driver for the USB-LCD-20x2 module in any programming language he wants. In the followings any package contents and formats will be presented separately with comments about usage mode.

- **Important Note:** This part of documentation is for advanced users only. All of the values are represented in hexadecimal format (i.e. 0x11). All type of messages contains in the first byte the message type. After the message type follows the message content. The **meaning** entry in description field always deals with the message content (first byte and byte 0 means the same first byte of content).

- **Important Note:** the message types are grouped regarding the running mode of the USB-LCD-20x2 module: KEYBOARD mode or FLASHER mode (when firmware upgrading will be possible). In each group were defined INPUT and OUTPUT type messages.

● ERROR message types

- **Important note:** the ERROR messages are commonly used and presented as the first group of messages. They are received only as response for OUT type messages.

RESULT_PARAMETER_MISSING 0x01

Request does not contain all necessary parameters (request is too short, no length or no address either).

RESULT_DATA_MISSING 0x02

Request attempts to write more data than was actually sent.

RESULT_BLOCK_READ_ONLY 0x03

Request attempts to erase or write a block within the flasher code section.

RESULT_BLOCK_NOT_ERASABLE 0x04

Request attempts to erase a block outside of "erasable" memory space.

RESULT_BLOCK_TOO_BIG 0x05

Request attempts to access more data at once than the HID buffer can handle.

RESULT_SECTION_OVERFLOW 0x06

Request (address & length) attempts to cross over a section boundary (section boundaries are set at 1Mbyte boundaries -> 0x100000, 0x200000... etc.).

● KEYBOARD mode INPUT type messages

IN_REPORT_POWER_STATE 0x01

Action: POWER key was pressed
Length: 1 byte (INPUT)
Meaning: a) At key press the value is always 1.
b) Using Windows OS this message is used by the OS (not included in the USBLCDServer.exe).

IN_REPORT_KEY_STATE 0x11

Action: key pressed
Length: 2 bytes (INPUT)
Data byte range: 0x00...0x0E
Meaning: a) Maximum two key presses in the same time will be accepted.
b) Nothing is pressed: 0x00, 0x00.
c) Single or two keys were pressed: the two values represents the two inputs code number.
d) Three or more keys pressed: 0x00, 0x00.

IN_REPORT_IR_DATA 0x21

Action: infrared signal activities received
Length: 1+ maximum 20 byte (INPUT)
Meaning: a) The first byte represents the length of the IR message (0x00...0x14)
b) The time interval between two adjacent level changing: two bytes represent one integer type value (LO byte first, HI byte last)

● KEYBOARD mode OUTPUT type messages

OUT_REPORT_LED_STATE 0x81

Action: setting the LED's status
Length: 1 byte (OUTPUT)
Data byte range: 0x00...0xFF
Meaning: a) Bit set to "1" switches the LED to ON.
b) Bit set to "0" switches the LED to OFF (every bit is representing one LED driver output).

OUT_REPORT_LCD_BACKLIGHT 0x91

Action: LCD backlight ON/OFF
Length: 1 byte (OUTPUT)
Data byte range: 0x00...0xFF
Meaning: a) 0x00 - Backlight OFF
b) 0x01...0xFF - Backlight ON

OUT_REPORT_LCD_CONTRAST 0x92

Action: setting the LCD contrast
Length: 1 byte (OUTPUT)
Data byte range: 0x00...0x40
Meaning: a) 0x00 means MAXIMUM contrast
b) 0x40 means MINIMUM contrast

OUT_REPORT_LCD_CONTROL 0x93

Action: LCD working mode
Length: 1 byte (OUTPUT)
Data byte range: 0x00...0x07
Meaning: a) Bit 0: cursor blinking (BLINK) ON ("1")/OFF ("0").
b) Bit 1: cursor switch ON ("1")/OFF ("0").
c) Bit 2: LCD switch ON ("1")/OFF ("0").

OUT_REPORT_LCD_CLEAR 0x94

Action: LCD screen clear
Length: 0 byte (OUTPUT)

OUT_REPORT_LCD_TEXT 0x98

Action: LCD write text
Length: 3 + 20 byte (OUTPUT)
Meaning: a) First byte: ROW (0x00...0x01)
b) Second byte: COLUMN (0x00...0x14)
c) Third byte: STRING LENGTH (0x00...0x14)
d) STRING - maximum 20 chars, in reduced ASCII code (see the LCD ASCII table)

OUT_REPORT_LCD_FONT 0x9C

Action: LCD font change (for the first 8 characters only which can be rewritten dynamically)
Length: 1 + 8 byte (OUTPUT)
Meaning: a) First byte: CHAR NUMBER (0x00...0x07)
b) BITMAP of the new chars (8*5 dots in 8 bytes, every byte holding row in the lower bits)

OUT_REPORT_EXIT_KEYBOARD 0xEF

Action: exit keyboard mode (switch to flasher mode)
Length: 2 byte (OUTPUT)
Meaning: a) Timeout in milliseconds between switching off the device and starting it in flasher mode (timeout=0-65535)
b) Byte 0: timeout & 0xff
c) Byte 1: (timeout >> 8) & 0xff

OUT_SET_SNOOZE_TIME 0xF8

Action: set up the timer value (this time is between switching off the device and witching it on in the new mode after the OUT_REPORT_EXIT_KEYBOARD message)
Length: 2 byte (OUTPUT)

- **Important Notes:** this message is not used anymore - use the snoozetime from OUT_REPORT_EXIT_KEYBOARD and OUT_REPORT_EXIT_FLASHER messages instead!

IN_REPORT_EXT_EE_DATA 0x31

Action: as response for OUT_REPORT_EXT_EE_READ and OUT_REPORT_EXT_EE_WRITE (check those messages for response description)

OUT_REPORT_EXT_EE_READ 0xA1

Action: read external EEPROM locations
Length: 3 byte (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 3: LENGTH of the read data (1-20)
Response:
- One of the ERROR_MESSAGES if error occurred
- If everything is ok: response type = IN_REPORT_EXT_EE_DATA
a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: LENGTH of the data what will be written (1-20)
d) DATA (LENGTH bytes)

OUT_REPORT_EXT_EE_WRITE 0xA2

Action: write external EEPROM locations
Length: 3 byte + LENGTH (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: LENGTH of the data what will be written (1-20)
d) DATA (LENGTH bytes)
Response:
- One of the ERROR_MESSAGES if error occurred
- If everything is ok: response type = IN_REPORT_EXT_EE_DATA in the same format as the request: a), b), c), d)

- **Important note:** if the received data does not match the sent one - a memory write error occurred!

IN_REPORT_INT_EE_DATA 0x32

Action: as response for OUT_REPORT_INT_EE_READ and OUT_REPORT_INT_EE_WRITE (check those message for response description)

OUT_REPORT_INT_EE_READ 0xA3

Action: read internal EEPROM locations
Length: 3 byte (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 3: LENGTH of the data what has to be read (1-20)
Response:
- One of the ERROR_MESSAGES (*) if error occurred

- If everything is ok: response type = IN_REPORT_INT_EE_DATA
  - a) Byte 0: addr & 0xff - ADDRESSLO
  - b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
  - c) Byte 2: LENGTH of the data what will be written (1-20)
  - d) DATA (LENGTH bytes)

OUT_REPORT_INT_EE_WRITE 0xA4

Action: write internal EEPROM locations
Length: 3 byte + LENGTH (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: LENGTH of the data what will be written (1-20)
d) DATA (LENGTH bytes)
Response:
- One of the ERROR_MESSAGES (*) if error occurred
- If everything is ok: response type = IN_REPORT_INT_EE_DATA in the same format as the request: a), b), c), d)

- **Important note:** if the received data does not match the sent one - a memory write error occurred!

OUT_REPORT_RELAY_ONOFF 0xB1

Action: turns on or off the mini switch on the microcontroller mainboard
Length: 2 byte (OUTPUT)
Meaning: Timeout in milliseconds until switching off the device from the moment when it turns on. When a command is received it will be executed immediatelly and the mini switch turns on.
If time value is 0x0000, it means instant "OFF", if the value is 0xFFFF, it means instant "ON" forewer (or until the next command received containing time value equal with 0x0000). Any values between extreme sets an n msec "ON" stage.
a) Byte 0: timeout LOW byte
b) Byte 1: timeout HIGH byte

● FLASHER mode OUTPUT type messages

OUT_REPORT_EXIT_FLASHER 0xFF

Action: exit flasher mode (switch to keyboard mode)
Length: 2 (OUTPUT)
Meaning: a) Timeout in milliseconds between switching off the device and starting it in keyboard mode (timeout=0-65535)
b) Byte 0: timeout & 0xff
c) Byte 1: (timeout >> 8) & 0xff

OUT_SET_SNOOZE_TIME 0xF8

Already described for keyboard mode (not used anymore)

OUT_REPORT_GET_VERSION 0xF1

Action: get the software version of HID (used also to reset the message queue of the HID after mode switch)
Length: 0 (OUTPUT)
Response: firmware version number in hexadecimal.

OUT_REPORT_ERASE_MEMORY 0xF2

Action: erase the main program memory location (anywhere inside the 64 byte-block)
Length: 3 (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: (addr >> 16) & 0xff - ADDRESSUP
Response:
- One of the ERROR_MESSAGES if error occurred
- If everything is ok:
Response type = OUT_REPORT_ERASE_MEMORY in the same format as the request: a), b), c).

OUT_REPORT_READ_MEMORY 0xF3

Action: read the main program memory location
Length: 4 (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: (addr >> 16) & 0xff - ADDRESSUP
d) Byte 3: LENGTH of the data what has to be read (1-32)

Response:

- One of the ERROR_MESSAGES if error occurred
- If everything is ok:

  Response type = OUT_REPORT_READ_MEMORY

  a) Byte 0: addr & 0xff - ADDRESSLO
  b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
  c) Byte 2: (addr >> 16) & 0xff - ADDRESSUP
  d) Byte 3: LENGTH of the data what will be written (1-32)
  e) DATA (LENGTH bytes)

OUT_REPORT_WRITE_MEMORY 0xF4

Action: write the main program memory location
Length: 4 + LENGTH (OUTPUT)
Meaning: a) Byte 0: addr & 0xff - ADDRESSLO
b) Byte 1: (addr >> 8) & 0xff - ADDRESSHI
c) Byte 2: (addr >> 16) & 0xff - ADDRESSUP
d) Byte 3: LENGTH of the data what will be written (1-32)
e) DATA (LENGTH bytes)

Response:

- One of the ERROR_MESSAGES if writing error occurred
- If everything is ok:

  Response type = OUT_REPORT_WRITE_MEMORY in same format as the request: a), b), c), d), e).

- **Important note:** if the received data does not match the sent one - a memory write error occurred!