

# jaLCDs TCP-IP-Interface documentation

v3.1

written by HooMair

1.0 General stuff

1.1 jaLCDs answers

2.0 Remote administration

3.0 Setting variables from external programs

4.0 Command reference

5.0 UDP-Port

6.0 Addon autostart

7.0 Stat-Reporting

7.1 The stat-reporting protocol

## **1.0 General stuff**

The TCP-IP-Interface can be used to send commands to jaLCDs (for example to change to the next screen, to disable/enable display output etc. – for a full list see 4.0 Command reference). External programs can define variables and fill them with values which can be used like any other predefined variable (\$name\$).

The interface is useful for remote administration via telnet-client, but it can also be used by programs to send data (even over a LAN) to jaLCDs which is displayed on the LCD. Using this, anyone can develop new functions which aren't included in jaLCDs.

There are 10 open ports for communication, range 9825-9834. One port can only be used by one program at a time, but all 10 ports can be used simultaneously to exchange data with jaLCDs.

jaLCDs answers – if the connection was established successfully – with two welcome lines which inform the client about the version number, port and ip. Then it waits for input. This input can be sent character by character or as a whole line, in any case it has to end with “CrLf” (meaning ASCII 13 & 10). jaLCDs will answer on any full line with a “statuscode” consisting of two digits and a text message, eventually followed by additional data (e.g. the config-list or the screen-list). The 2 digits in front of the statuscode can be easily processed in external programs (the same concept is used by the FTP-protocol).

jaLCDs also sometimes sends information to all connected clients...one event that causes this is the change of the config, this causes jaLCDs to send a “configchange:[newconfigfile]” to all clients. Another event is the end of jaLCDs, so if the user closes the program jaLCDs sends a “terminate” to all connected programs.

### **1.1 jaLCDs answers**

<b>code</b>	
99	Command not understood
10	Authentication has been successful (isn't of any importance since the need to authenticate has been removed from jaLCDs, but is being kept for compatibility reasons)
20	Command has been executed. Eventually the next lines will continue the result of the command.
21	Command wasn't executed because something is missing (e.g. a config file or a screen)
30	The syntax is wrong.

## **2.0 Remote administration**

It's best to do remote administration tasks with a telnet client. jaLCDs has been successfully tested with the standard windows telnet client (the one you get when entering "telnet address port" at the dos prompt) and CRT v3.4 but should work with other clients, too.

**IMPORTANT: Since v2.5, no more authentication (using the "authme" command) is needed to do remote administration tasks.** For compatibility to older programs, the command is still understood.

### **3.0 Setting variables from external programs**

The TCP-IP-Interface can be used by external programs to define custom variables in jaLCDs which can be displayed on the display. This is as simple as possible for the user – he just uses the custom variables like the ones defined by jaLCDs (for example \$variablename\$).

There are a few restrictions on variable names – they can be as long as you wish, but mustn't include '\$', '\$' or spaces.

The command to define a variable is „setvar“ and is used like this:

```
setvar [name] [value]
```

Value has to be a string of any length, name is the name of the variable which the user has to use to display the variable.

Attention: Variable names are case-sensitive!

There can be up to 200 custom variables in jaLCDs at any time.

All variables and their values are deleted when jaLCDs is being closed.

You can download a sample program (fully commented source in VB6) at <http://www.jalcds.de>

**Since v2.5, you can also use the udp-port to define variables. More about this in 5.0**

## **4.0 Command reference**

This reference contains the supported commands of jaLCDs v3.0. Later versions maybe support even more commands, please check the version history on <http://www.jalcds.de>  
As soon as a new version with new commands is released, an updated version of this document will be created.

### **Setvariables – setvar**

This command can be used to create new custom variables or to set the value of an already existing custom variable.

syntax:            setvar [ name] [value]

parameters:       name: The name of the variable  
                  value: A string which should be saved in the named variable

answers:           20 Executed  
                  30 Syntax error

special info:      The restrictions for variable names have to be followed

### **Closeconnection – exit**

This causes jaLCDs to close the connection between jaLCDs and the client program.

syntax:            exit

parameters:       none

answers:           20 Logout successful

special info:      Just closing the connection on the client side has the same effect as the exit-command, though using exit to close a connection is the better way.

### **Exit jaLCDs– shutdown**

jaLCDs will close itself without asking for confirmation.

syntax:            shutdown

parameters:       none

answers:           20 Executed

### **Screencontrol – screen**

The screen command can be used to get the name of the currently shown screen, but also to have jaLCDs jump to a screen. If it's used without parameters, the name of the shown screen will be returned. Otherwise it'll try to jump to the screen defined in the parameter.

syntax:            screen[screenname]

parameters:       screenname: Name of the target screen.

answers:           20 Currently shown: [screenname]  
                    20 Screen changed successfully  
                    21 Screen not found  
                    30 Syntax error

special info:      Screen names are case-sensitive!

### **List of all screens – listscreen**

This command returns a list of all screens in the current configuration.

syntax:            listscreen

parameters:       none

answers:           20 Executed (followed by the screen list)

### **LCD output control – lcdoutput**

This can be used to control whether jaLCDs is allowed to send the generated screens to the LCD or not (screens are still being generated, even if output is switched off). If this is used without parameters, the answer contains the current state.

syntax:            lcdoutput [on/off]

parameters:       on/off: if set to „on“, jaLCDs will output screens to the lcd, otherwise it won't.

answers:           20 Output has been activated  
                    20 Output has been deactivated  
                    20 Output is activated  
                    20 Output is deactivated  
                    30 Syntax error

### **Clearscreen – lcdclearscreen**

This causes the LCD to clear the screen.

syntax:            lcdclearscreen

parameters:       none

answers:           20 Executed

special info:      If this is used and output stays activated, the empty screen won't stay empty for a long time...

### **Display row – lcddisplayline**

This command displays a text line on the display.

syntax:            lcddisplayline [row] [text]

parameters:       row: can be 1 to 4 – this controls the row where the text should be displayed  
text: this doesn't need any further explanation, I hope ;-)

answers:           20 Executed  
                    30 Syntax error

special info:      see special info of lcdclearscreen

### **Display a screen for a given time – lcddisplayscreen**

This shows a screen for a given time, the normal screens will continue in the background but won't be shown.

syntax:            lcddisplayscreen [time] [screen]

parameters:       time: Display time in milliseconds  
screen: String containing the whole screen (line 1, 2, 3 and 4)

answers:           20 Executed  
                    21 The text length doesn't match the screen size

special info:      The parameter „screen“ has to contain exactly enough characters to fill a whole screen, not more, not less!

### **Display a screen after the currently displayed one has reached its time limit – lcddisplaydelayd**

This works like lcddisplayscreen, but waits for the next screen change.

syntax:            lcddisplaydelayd [time] [screen]

parameters:       time: Display time in milliseconds  
screen: String containing the whole screen (line 1, 2, 3 and 4)

answers:           20 Executed  
                    21 The text length doesn't match the screen size

### **Defining a custom char on the LCD – lcdsetchar**

This can be used to store a custom character in one of the eight character slots.

syntax:                lcdsetchar [slot] [byte1] [byte2] [byte3] [byte4] [byte5] [byte6] [byte7] [byte8]

parameters:        slot: Number between 1 and 8 – defines the slot in which the char should be saved.  
                      byte1-byte8: 8 Bytes which contain the information about the character. More about this can be found on websites about the HD44780 controller.

### **Requesting the size of the LCD – lcdsize**

The response contains the length in chars.

syntax:                lcdsize

parameters:        none

### **Config control – config**

This command can be used to load a different configuration. If no parameters are given, the answer contains the current config name.

syntax:                config [configname]

parameters:        configname: Name of the config (filename without .cfg)

answers:              20 Executed  
                      21 Config not found  
                      30 Syntax error  
                      20 Current config: [configname]

### **List of all configs – listconfig**

This will return a list of all available configs (meaning all .cfg-files in the jaLCDs-directory)

syntax:                listconfig

parameters:        none

answers:              20 Executed (followed by the list)

### **Add a screen to the current config – addscreen**

This will add a screen with the given parameters. The added screen won't be saved in the config file, so reloading the config or restarting jaLCDs will delete it!

syntax:            addscreen [name] [displaytime] [updateinterval] [scrollspeed] [growspeed] [in] [out]  
[line 1]<br>[line 2]<br>[line 3]<br>[line 4]

parameters:       name: Screenname  
displaytime/updateinterval/scrollspeed/growspeed: Take a look at your own configs to see what this is :)  
in/out: name of the in/out-transition  
line x: The 4 lines of the screen, separated by <br> (similar to HTML)

answers:           20 Executed  
                    30 Syntax error

If you don't want any transitions, use „none“ as name of the transition.

### **Deletethelast addedscreen - deletescreen**

You can delete screens you added with addscreen using this command. You can only delete the last screen you added, but maybe this will be changed in future versions.

syntax:            deletescreen

parameters:       none

answers:           20 Executed  
                    21 You can't delete screens defined in the config-file

### **Parseascreen - lcdparsescreen**

This command shows a screen on the display just like lcd displayscreen, but the screen will be parsed, meaning all variables will be replaced and all commands will be executed just as if it was a normal screen.

syntax:            lcdparsescreen [displaytime] [updateinterval] [scrollspeed] [growspeed] [mode]  
[[line 1]<br>[line 2]<br>[line 3]<br>[line 4]

parameters:       displaytime/updateintervall/scrollspeed/growspeed: I hope this doesn't need any explanation.  
mode: If this is set to 1, the screen will be shown immediately. If it's set to 0 it will be shown at the next regular screen change.

line x: The 4 lines of the screen, separated by <br> (similar to HTML)

answers:           20 Executed  
                    30 Syntax error

### **Blockthescreenchange - blockscreenchange**

This can be used to block parts of the internal screen generation mechanism. It's best to use this together with lcdoutput, this will prevent the little „stops“ jaLCDs sometimes creates even though output is switched off (caused by the screen transitions, which are still being generated)

syntax:            blockscreenchange on/off

parameters:       on/off: Activates/deactivates the block

answers:           20 Executed

### **Get the value of a variable (>have jaLCDs parse a query) - getvar**

The command “getvar” can be used to get the current value of a variable. Everything after the initial “getvar” command and the space character is being parsed by jaLCDs as if it was a line on a screen and the result is being sent back. Therefore you are not limited to ask for single variables but you can have jaLCDs parse complex queries containing for example  $math:::$-commands etc.$

syntax:            getvar [query]

parameters:       query: This is the query string that’s being parsed and sent back by jaLCDs

answers:           20 Value: [answer]

### **Reload char files - reloadchars**

This command causes jaLCDs to search for char-files in its home directory and to reload them (the same happens if “reload chars” is selected from the jaLCDs popup menu)

syntax:            reloadchars

parameters:       none

answers:           20 executed

### **Refresh information about e-mail accounts - refreshmails**

This causes jaLCDs to refresh the information of all e-mail accounts.

syntax:            refreshmails

parameters:       none

answers:           20 executed

## **5.0 UDP-Port**

You can also send your variables to jaLCDs over the UDP protocol. This doesn't need any connections and is therefore easier to use. You can only use the commands "setvar" and "lcdoutput on/off" over UDP.

UDP is useful for external programs which only want to set variables – they don't need to establish a connection to a port which makes programming much easier.

You have to send your commands to port 9835, and don't wonder if you don't get any response to your command, you won't get this over udp!

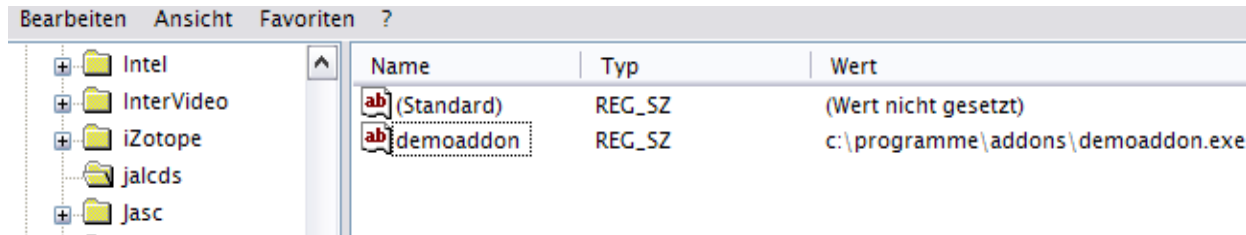
## 6.0 Addon autostart

Addons can be automatically started by jaLCDs. jaLCDs checks the registry key

HKEY\_CURRENT\_USER\Software\jalcds

for paths to addon programs. 10 seconds after jaLCDs has been started, the programs defined in this key are started by jaLCDs.

To add a program there, just add a new value of the type "REG\_SZ". The name is irrelevant, the value has to be the path to your program. In regedit, this would look like this:



## 7.0 Stat-Reporting

Stat-Reporting is a feature which enables users to upload variables to a web server and to have the server process this data.

I have written an own script to manage and process these transmitted variable values which can be used by anybody without needing knowledge on programming etc. and without needing a web server. This script and further information about it can be found in the community area at <http://www.jalcds.de> (look for “stats center”).

For people interested in processing the variables with own scripts, I have created the possibility to enter the URL where the data is being sent to. Add the command line parameter “statsurl=[url]” and replace the [url] with the full URL to your script.

### 7.1 The stat-reporting protocol

The data is transmitted with a HTTP POST command and usually looks like this:

```
user=[username]&pass=[password]&query=cmFtdXNlZA==MzYzICAgICAgICAgICAgICAgICA=Y3B1bG9hZA==MlTAwICAgICAgICAgICAgICAgICA=bWFpbDE=$NSAgICAgICAgICAgICAgICAgICA=bWFpbDI=$MCAgICAgICAgICAgICAgICAgICA=dXBzcGVlZA==MCw3ICAgICAgICAgICAgICAgICA=ZG93bnNwZWVkb$MCw3ICAgICAgICAgICAgICAgICA=dXBzcGRyb3V0ZXI=$MCwyICAgICAgICAgICAgICAgICA=ZG93bnNwZlJvdXRlcmg==MCwyICAgICAgICAgICAgICAgICA=
```

[username] and [password] are the username and password entered in the stat reporting configuration and can be used to distinguish between multiple persons or instances of jaLCDs and therefore create multi-user-capable processing scripts (like it’s done on <http://www.jalcds.de>). Everything after “query=” belongs to the data part and contains the variable values.

This whole data string has to be splitted into multiple parts during processing:

```
cmFtdXNlZA==MzYzICAgICAgICAgICAgICAgICA=Y3B1bG9hZA==MlTAwICAgICAgICAgICAgICAgICA=bWFpbDE=$NSAgICAgICAgICAgICAgICAgICA=bWFpbDI=$MCAgICAgICAgICAgICAgICAgICA=dXBzcGVlZA==MCw3ICAgICAgICAgICAgICAgICA=ZG93bnNwZWVkb$MCw3ICAgICAgICAgICAgICAgICA=gICA=dXBzcGRyb3V0ZXI=$MCwyICAgICAgICAgICAgICAgICA=ZG93bnNwZlJvdXRlcmg==MCwyICAgICAgICAgICAgICAgICA=
```

The **red** parts are encoded variable names  
The **blue** parts are the encoded variable values  
The remaining black parts are separators

So in the end the structure looks like this:

**[variable name]**\$(**[variable value]**)

This is being repeated for each variable like shown in the above sample.

The encoding mechanism used is simple base64-encoding, so you have to base64-decode the parts **one by one** after separating them. Don’t decode the whole string at once!