# USER MANUAL
# OVERVIEW

---

## Avisaro 2.0 Product Series

Data Logger Box (SD)

WLAN Device Server (2.0)

Ethernet / TCP
Logger Box (SD)

Data Logger Module (SD)

WLAN Module (2.0)

Other configurations

Version 1.2
2008.02.12

# THIS DOCUMENT

## HISTORY

| | | |
|---|---|---|
| 2008-01-01 | | Initial version |
| 2008-02-07 | | Added CAN commands table |
| 2008-02-12 | | Added Ethernet (TCP und raw) commands |

## REQUIRED FIRMWARE

This document describes firmware version 1-16 or higher.

Navigate to the relevant product page on www.avisaro.com to download latest firmware versions.

## CONTACT

Avisaro AG
Vahrenwalderstr. 7 (tch)
30165 Hannover

Germany

Telephone:
+49-511-7809390
Telefax:
+49-511-35319624

eMail:
support@avisaro.com

# TABLE OF CONTENT

## ABOUT THIS DOCUMENT

This document provides a compact introduction into the functions and features of the Avisaro 2.0 Product series. Since all Series 2.0 products are similar in how to use them, this document is "one doc fits all".

Further detailed documentation is available:

| **Document** | **Description** | **Where to find** |
|---|---|---|
| Avisaro Command Interface | Lists all commands and their syntax. Describes packet format and other details | www.avisaro.com (Download section) |
| Avisaro Basic Scripting | Lists all BASIC commands and their syntax. Comes with example listings. | www.avisaro.com (Download section) |
| Avisaro Hardware Documentation | Describes PIN layout, electrical and mechanical dimensions of the products. | www.avisaro.com (Download section) |

# CONNECT AND POWER UP

## BOX: RS232 INTERFACE

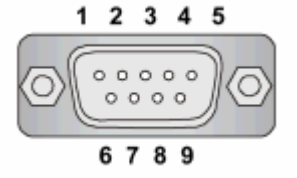
Relevant products are:

Data Logger Box (SD) with RS232 interfaces

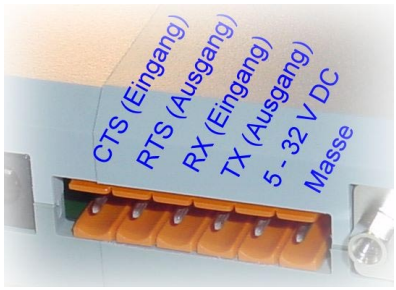
WLAN Device Server (2.0) with RS232 interfaces

Depending on the interface purchased, the Box comes with a SubD connector or with a WAGO cage clamp connector. The SubD (male) connector conforms to the standard pin layout:

1 Data Carrier Detect (DCD)
2 Receive (RxD) Data going to Avisaro Box
3 Transmit (TxD) Data going to device
4 Data Terminal Ready (DTR) Avisaro Box is up
5 Signal Ground GND
6 Data Set Ready (DSR) Client is up and running
7 Request To Send (RTS) Avisaro wants to send data
8 Clear To Send (CTS) Client is ready to receive data
9 Not used (Ring Indicator (RI)




If the box comes with a WAGO connector, the pin layout is as follows:




The barrel connector is layouted as follows. Supply voltage is 6 – 32 V. Diameter of the connector is 2.1mm:

## BOX: CAN INTERFACE

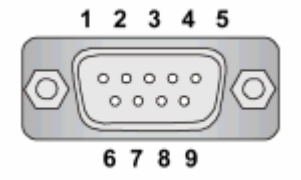
Relevant products are.

Data Logger Box (SD) with CAN interfaces

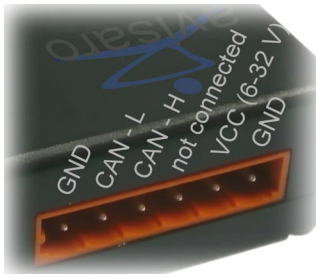
WLAN Device Server (2.0) with CAN interfaces

Depending on the interface purchased, the Box comes with a SubD connector or with a WAGO cage clamp connector. The SubD (male) connector conforms to the standard pin layout:

1 not connected
2 CAN-L
3 GND
4 not connected
5 not connected
6 GND
7 CAN-H
8 not connected
9 not connected




If the box comes with a WAGO connector, the pin layout is as follows:




The barrel connector is layouted as follows. Supply voltage is 6 – 32 V. Diameter of the connector is 2.1mm:

## BOX: ETHERNET (TCP / RAW) INTERFACE

The Ethernet connection serves for four functions:

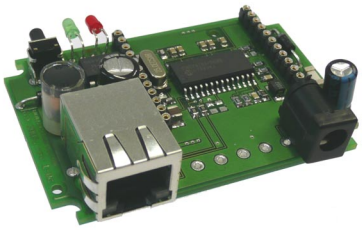
1) **Data interface with a TCP/IP connection:** Connect with a TCP connection to the Avisaro product (i.e. 192.168.0.73 on port 23). Using this TCP connection, all commands can be entered.

2) **Network interface for user data (i.e. data logger):** Within BASIC scripting or in combination with other data interfaces (like RS232, CAN, …) the Ethernet network interface can be used for user defined networking (outgoing/incoming TCP and UDP connections). Example: Automatic data logger (all data through this TCP connection is written automatically into a file). See BASIC scripting for details.

3) **Data interface with Ethernet type 1 frames:** A low level communication using Ethernet type 1 frames is accepted. Avisaro accepts Ethernet packets its MAC address.

4) **Network interface for web server:** For easy configuration, the Avisaro products have build in web server which provides pages for configuration. Default IP address is 192.168.0.73 / default user and password for web site is: admin / 1234

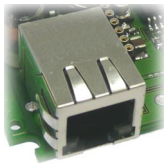
Data Logger Box (SD) with Ethernet interfaces and WAGO cage clamp for power



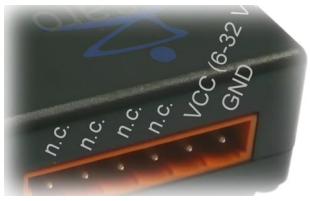
Data Logger Box (SD) with Ethernet interfaces and barrel connector (shown without enclosure)

Ethernet interface is a standard RJ45 jack:

Standard Ethernet Layout



If the box comes with a WAGO connector, power is supplied using this connector




The barrel connector is layouted as follows. Supply voltage is 6 – 32 V. Diameter of the connector is 2.1mm:

## BASE AND TRAILOR MODULES

The Base and Trailor Modules are the building blocks for all Series 2.0 products. The modules are designed to be integrated into other products to become a part of their.

Data Logger Module (SD) with RS232, CAN, I2C, SPI and other interfaces

WLAN Module (2.0) with RS232, CAN, I2C, SPI and other interfaces

Other configurations

### *INTERFACE (RS232, I2C, SPI, …) SELECTION*

The active interface (RS232, SPI, I2C, CAN, …) is configurable. The default configuration is RS232. See next chapter "Configuration" on page 10 for how to change this configuration (either through auto run file on SD card, web interface or data interface. I.e. "prot i2c" for I2C interface to become active).

### *PIN LAYOUT AND SIGNAL LEVELS*

The module is powered with 3.3V. Higher supply voltage will damage the product permanently. Signal levels, however, can be up to 5V.

All data interface do not contain any driver circuitry. Thus: **RS232** is 3.3V level. To operate with a PC or other device, a MAX232 IC is needed. **CAN** is designed as Tx and RX – a CAN transceiver IC is need to operate on a CAN bus.

Refer to the hardware documentation for details on the pin layout. As an example, the pin layout of the module when configured as RS232 is displayed here. When configured as i.e. CAN, pin layout changes. Power (pin 24 and pin 13) remain always power pins.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | P | VBAT | 1 | 24 | VCC (3.3V) | P | 24 |
| 2 | O | LED1 | 2 | 23 | IN_CS0 | O | 23 |
| 3 | O | LED2 | 3 | 22 | IN_CS1 | O | 22 |
| 4 | I | KEY | 4 | 21 | IN_CS2 | O | 21 |
| 5 | I | DCD | 5 | 20 | IN_CS3 | O | 20 |
| 6 | I | DSR | 6 | 19 | IN_SCK | O | 19 |
| 7 | O | DTR | 7 | 18 | IN_MOSI | O | 18 |
| 8 | I | RING | 8 | 17 | IN_MISO | I | 17 |
| 9 | O | TXD | 9 | 16 | nc | | 16 |
| 10 | I | RXD | 10 | 15 | nc | | 15 |
| 11 | I | CTS | 11 | 14 | nc | | 14 |
| 12 | O | RTS | 12 | 13 | GND | P | 13 |

*'P': Power, 'I': Input, 'O': Output, 'B': Bidirectional*

VBAT: Battery Supply for the real time clock. See chapter … for details

LED1: Module is up and running

LED2: Module is working (transmitting data)

KEY: Stop current action. Start defined action (data logging, WLAN transmittion)

TXD, RXD, ….: RS232 connection

IN_*: Internal SPI bus to connect trailer modules

## WATCH OUT

The Avisaro 2.0 Products are high tec products designed to be used by experts. Even though their design is robust to withstand electrical and mechanical hazards, all products should be handled with care.

When products are not operated as designed, there is risk of damage to the products and to other products around.

# CONFIGURATION

There are three ways to configure the Avisaro products.

1) **Configuration using SD memory cards**: All data logging products can be configured using a configuration file on a memory card. Place all configuration commands (see next paragraphs for examples and separate document "Command Interface Manual") in a file called "autorun.txt". Insert card and power on module. This will always work even if all other methods are blocked i.e. by faulty baud rate parameter.
   Got no SD Card slot on the product ? See chapter "Building blocks" on page 14 on how to add one temporarily.

2) **Configuration using Web-Interface:** All products with a WLAN or LAN interface can be configured using a build in web interface. Connect your PC to the modules WLAN or LAN. Type in the IP Address of the Avisaro module into the browsers address line. All settings are now set by the web interface. Default user and password for web site is: admin / 1234 .

3) **Configuration using Data Interface:** All commands can be typed or send over the data interface (RS232, SPI, CAN, …). Commands can be send in ASCII format or in binary packed format (see "Command Interface vs. Packet API vs Scripting" on page 12). See separate document "Command Interface Manual" for the syntax of the commands.

## DEFAULT VALUES FOR DATA INTERFACE

When shipped, the products are pre configured ("Default Settings") depending on the interface activated. Modules are preconfigured with the RS232 connection activated:

1) **RS232:** Default setting is 9600, n, 1 and no flow control

2) **I2C:** Default bus address is decimal 73

3) **SPI:** No special setting necessary

4) **CAN:** Default baudrate is 125 kbit/s. Default CAN ID is decimal 73

5) **Ethernet / TCP:** The data interface can be reached at ip address 192.168.0.73 on port 23.

6) **Ethernet / raw:** The default MAC address is printed on the product

## DEFAULT VALUES FOR NETWORK INTERFACE

The network interface must be active for Ethernet TCP / raw to work. The network interface is found automatically upon power up. If WLAN and LAN are attached, the WLAN becomes the network interface. Only one network interface can be active as a network.

1) **WLAN setting:** In default mode, the Avisaro products creates an adhoc WLAN network with the SSID name "avisaro". Do a network search with your PC and it will show up in the list of available networks. Connect to it. Set the PCs IP address to 192.168.0.xx and the mask to 255.255.255.0. Type in the default Avisaro IP address 192.168.0.73 in a browser to connect to the module. Default user and password for web site is: admin / 1234 .

2) **LAN setting:** Connect via network cable. Set the PCs IP address to 192.168.0.xx and the mask to 255.255.255.0. Type in the default Avisaro IP address 192.168.0.73 in a browser to connect to the module.

## EXAMPLE CONFIGURATION: RS232

```
prot rs232                  Activate RS232 interface
rs232 115200 8 N 1 NONE     Set baudrate and other settings
```

## EXAMPLE CONFIGURAITON: CAN

```
prot can                    Activates CAN  as data interface
CAN 125000 49 0 49 0        Set baudrate and other settings
```

## EXAMPLE CONFIGURATION: ETHERNET (TCP)

The following commands are used to configure the Avisaro product to accept an incoming TCP connection. All settings shown here are default – change only if required:

```
net eth                     Network interface to Ethernet
prot sock                   Activates TCP as data interface
sockio 23                   Listen on port 23
ip local 192.168.0.73       Set modules IP address (for TCP and Web)
```

## EXAMPLE CONFIGURATION: ETHERNET (RAW)

The following commands set the "raw ethernet" interface to receive packets addressed to mac address 02020304050a coming from mac address 0019dbb5c231. TCP settings are not necessary since low level Ethernet is used.

```
net eth                                 Network interface to Ethernet
eth 02020304050a                        Activates raw Ethernet  as data interface
ethio 0019dbb5c231 ffffffffffff off off 10   Set listen MAC and filter
```

## COMMAND INTERFACE VS. PACKET API VS SCRIPTING

There are three modes to operate or to interact with the module:

1) The **Command Interface** allows to type and send commands in ASCII format. For example, a RS232 product is connect with a terminal program on a PC, one can communicate by typing in commands and by reading the answer on the terminal.

2) The **Packet API** is designed for programmable devices (microcontroller) to control the Avisaro product. Commands and data are packaged in simple frames. This is a powerful and fast way to communicate with the module. The frames contain binary data elements.

3) **Scripting** is used to customize the behaviour of the module and to implement stand alone functionality. As programming language, a BASIC like syntax is used. That is simple to understand and it does not need a special programming environment. Typical application would be data logging: receive data from one of the interfaces, format the data and store it on a memory card or send it via (wireless) network.

### COMMAND INTERFACE

The command interface is designed to communicate with the module 'naturally'. When the RS232 interface is active and connected with a PC, one can type in the commands on a regular terminal programm (such as Hyperterminal which comes with Windows). Other interfaces work the same.

There is a long list of commands to configure and to operate the Box or the Module (see "Attachment: Commands" on page 18 and separate document "Command Interface Documentation")

### PACKET API

The packet interface uses a simple frame format for communication. With "Internal Portnumbers" multiple communication – such as several opened files or several TCP/IP connections – at the same time are possible. See separate document "Command Interface Documentation".

| Header Byte | Length MSB | LSB | Internal Port(*) | Payload | CRC MSB | LSB |
|---|---|---|---|---|---|---|

*Avisaro Packet API frame format.* *(* only if frame is a data frame*

### SCRIPTING

Scripting is described in its own chapter. See "BASIC Scripting" on page 13.

# BASIC SCRIPTING

The Avisaro Base Module has a build in BASIC programming language. It is designed to program small applications such as data logger or device server functionality. It is not designed to write large, complex applications.

The BASIC program is stored in the internal flash of the Avisaro Module – thus it needs to be loaded only once. By setting the autorun flag (see ) the BASIC program starts automatically upon power up. It is a simple yet very powerful feature.

Browse to www.avisaro.com and navigate to the "Anwendung" or "Application" section for examples.

## PROGRAMMING ENVIRONMENT

Use your preferred text editor to type the BASIC program. There is no special programming environment since applications are not complex. For an easier start, use one of the example programs and modify it. The source code can be 4 kByte large (with 2.8 kByte tokenised code) and 8 kByte space for variables, arrays and other space needed at runtime.

## LOADING

The BASIC program can be uploaded three ways:

1) Using the data interface (i.e. RS232, I2C):
   Type the command "load" and <enter>. Than upload the textfile with the BASIC program. Most terminal program support the option "send textfile" to do that. Finish upload by typing in "+++" (=stopsequence). It is usefull to have the three +++ at the end of the textfile.

2) Using a SD memory card:
   Copy the textfile (i.e. "basic.txt") with the BASIC program to a SD card. Create a second textfile called "autorun.txt" on this SD card. Enter the following command in this file: "load basic.txt". Insert a newline after this command (<enter>). Place SD card into module and power up. The BASCI program will be loaded into memory.

3) Using the Web Interface:
   With a network interface (WLAN or LAN) attached, the upload can be done using the Webinterface. Browse to the module's home page and navigate to the BASIC page. Select the file to upload and press submit.

## EXECUTION

To execute a BASIC Script, the "run" command is issued. Using the "run auto" command, the autostart flag is set. When the Box/Module is powered up, the script is started automatically. A single "run" starts the script once. The run command can be issued by the data interface, autorun.txt file on SD card and via web interface.

How to stop execution depends somewhat on the design. The BASIC script can terminate itself – it was programmed to do so. If it was started with "run" without any flags, a power off/on cycle will stop execution. If the "run auto" flag is set, termination can be done two ways: using a SD card with the "autorun.txt" file containing a "run manual" command or using the Web interface (if WLAN/LAN is present).

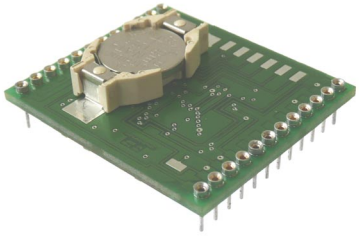
# BUILDING BLOCKS

Avisaro 2.0 Product Series is designed to mix and match. Just like LEGO building blocks, most elements can be connected to get the functionality one needs. Need a CAN Data Logger with WLAN access ?

## BASE MODULE

The Avisaro 2.0 Base Module contains the main processing power to drive the WLAN or SD-Card trailer modules. The Base Module is designed to piggyback the Trailer Modules and is designed to fit on the connector boards.

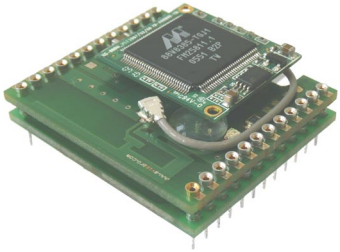
*Processor and other circuitry*



*optional on board battery for real time clock (a pin for Vbat is available, too)*

## TRAILER MODULES

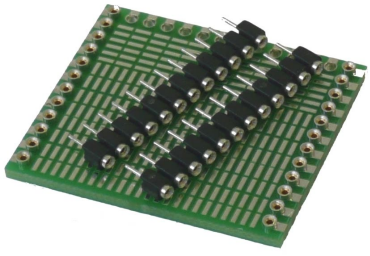
Trailer modules bring WLAN, SD card slot and other functionality.



*WLAN Trailer module, piggybacked with the Base Module*

*SD-Card Trailer module piggybacked with the Base Module*

*All modules stacked together*

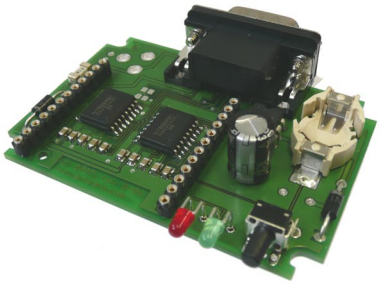


*Experimental PCB with connector strips for height spacing*

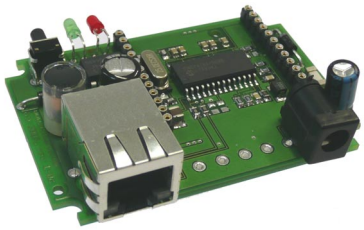
## CONNECTOR BOARDS

Connector boards contain interface connectors, driver circuitry such as MAX232 and CAN transceivers and power supply circuitry. A battery holder to power the real time clock is on board as well.

*CAN Board with RTC battery*



*RS232 Board (no Base and Trailer module stacked on)*



*Ethernet Board (no Base and Trailer module stacked on)*

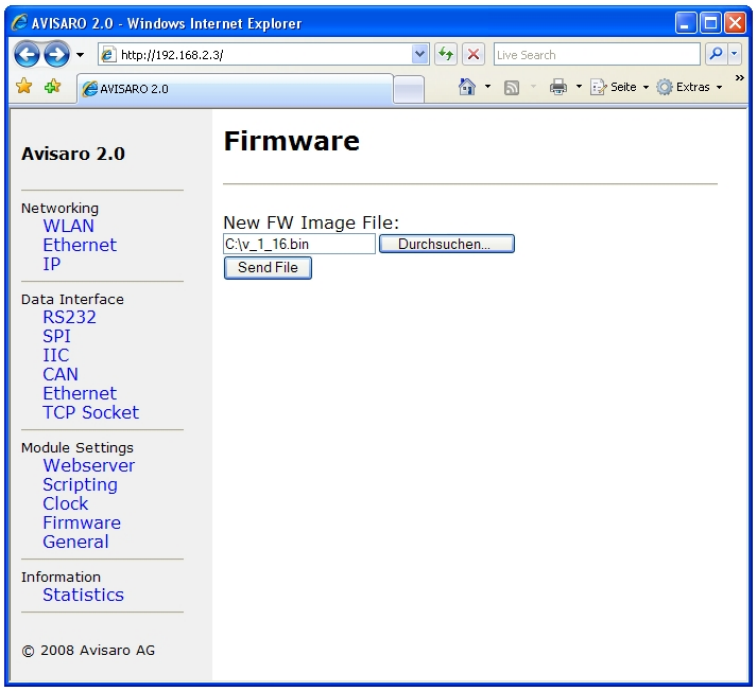
# FIRMWARE UPDATE

The firmware of the Avisaro Base Module can be updated to add newly released features or to correct bugs. There are three ways to update the firmware: 1) Using the data interface (i.e. RS232) 2) using a SD memory card 3) using the web interface.

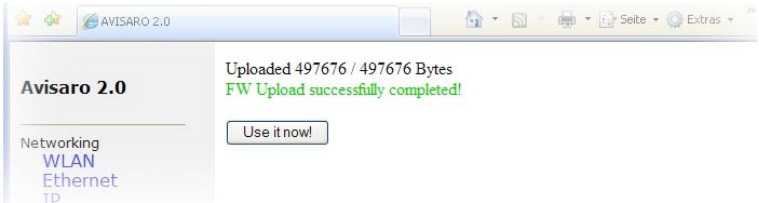
**Attention: Do not interrupt a firmware update process. Make sure power supply is stable. A firmware update always has the risk of leaving the module in a non-functional state.**

## FIRMWARE UPDATE USING WEB INTERFACE

If a network interface is available (WLAN or LAN) and activated, a new firmware can be upload using the web interface. Navigate to the "Firmware" section and select the firmware file.




Press the "Send File" button and wait for the response. It make a couple (~10 sec) of seconds to upload the file.




If the upload was successful, the "use it now!" button allows to start the actual update process. Click to activate.

## FIRMWARE UPDATE USING A SD MEMORY CARD

To perform a firmware update using a SD memory card, follow the steps:

1) Copy the firmware image on a SD memory card. The name of the firmware typically looks like "avi_v1-9.bin". You find the latest version on the www.avisaro.com home page. Usually, the downloaded firmware is zipped – use your preferred unzip program.'
   Create a file "autorun.txt" on the same SD card. This file contains commands to be executed after power up. Fill this file with the following commands:
   loadfw avi_v1-9.bin
   progfw

2) Switch off the Avisaro module. Insert the SD card.

3) Switch on the Avisaro module. The process will be started by first loading the firmware into the module (loadfw) and than performing the firmware update (progfw). The process can be supervised by observing the LEDs on the base module. A flashing red LED signals a ongoing process. A solid red and green LED signals finished process.

4) Switch off the module, take out SD Card. Remove the files to avoid an accidental firmware update.

Alternatively to using the autorun.txt file, the commands can be entered using the command line interface.

## FIRMWARE UPDATE USING DATA INTERFACE

Firmware update using the data interface requires to use the packet interface.

1) Issue the loadfw command with no parameter

2) Transmit the firmware image (packed in frames)

3) Issue the progfw command

# ATTACHMENT: COMMANDS

| Command | Parameter | Description | | Binary |
|---|---|---|---|---|

## *INTERFACE CONTROL*

| | | | | |
|---|---|---|---|---|
| PROT | <> | Selects data interface configuration (RS232, SPI, I2C, …., IOs ) | | 0x10 |
| I2C, I2C? | <address> | Sets I2C slave address | | 0x11 |
| SPI, SPI? | <><> | Sets the SPI parameters | | 0x12 |
| RS232, RS232? | <><><><> | Sets RS232 parameters | | 0x13 |
| CAN, CAN? | <> <> <> <> | Set CAN paramenters | | |
| CSTAT? | | Prints CAN statistics | | |
| NET | <WLAN / ETH / AUTO, NONE> | Controls the network interface | | |
| NET? | | Prints active network interface | | |
| ETH | <address> | Sets MAC address | | |
| ETH? | | Prints MAC address | | |
| ETHIO | <><><> | Configures Ethernet as low level data interface | | |
| ETHIO? | | Shows all settings of the Ethernet | | |
| ECHO | <text> | Echoes back user input | | 0x14 |
| SOCKIO | <port> | Sets a new port number for the TCP socket I/O protocol | | |
| SOCKIO? | | Shows port number used by TCP socket I/O protocol | | |

## *GENERAL COMMANDS*

| | | | | |
|---|---|---|---|---|
| CMDS? | | Shows all commands | | 0x41 |
| VER? | | Prints firmware version | | |
| NAME? | | Prints module name | | |
| NAME | <name (max 16)> | Sets module name | | |
| RESTART | | Reboot the module | | 0x43 |
| STPSEQ | <> | Changes the stop sequence | | 0x44 |
| STPSEQ? | | Shows the stop sequence | | |
| PROMPT | | Sets new prompt string, zero means no prompt | | 0x45 |
| TIME | <><><><> | Sets RTC time values | | 0x46 |
| TIME? | | Show RTC time | | 0x47 |
| ERR? | | Prints textual error | | 0x48 |
| ERRORS? | | Shows all errors | | 0x49 |

## *WLAN / LAN INTERFACE*

| | | | | |
|---|---|---|---|---|
| IP | LOCAL <IP address><br>GW <IP address><br>MASK <IP address><br>DNS <IP address><br>DCHP <IP address> | Sets IP address of module itself, gateway, mask and DNS server | | |
| IP | DHCP <on/off> | Switches DHCP on or off | | |
| IP? | | Prints all IP settings | | |
| WLAN? | | Prints all WLAN settings | | |
| WLAN | SSID <name> | Sets WLAN network name | | |
| WLAN | MODE <INFRA/IBSS> | Ad-hoc or infrastructure mode | | |
| WLAN | CHANNEL <number> | Sets channel number | | |
| WLAN | SECURITY | Controls WLAN Security | | |

| | | | | |
|---|---|---|---|---|
| | <NONE, WEP40, WEP104, WPAPSK> | | | |
| WLAN | PASS <passphrase> | Sets WPA Passphrase | | |
| WLAN | WEP <key> | Sets WEP Key | | |
| WLAN | PS <on/off> | Controls WLAN powersave | | |
| HTTP | ON, OFF, USER, PASS | Changes setting of web server | | |
| HTTP? | | Prints settings of web server | | |
| PING | <IP address> | Issues a ping command | | |
| DNS | <Internet address> | Gets IP address for domain | | |
| CONNECT | <handle> <IP address> <port> <tx timeout> <wait> | Creates a TCP connection | | |
| UDP | <handle> <IP address> <port rx> <port tx> <tx timeout> <check> | Creates a UDP socket | | |
| LISTEN | <handle > <port> <tx timeout> <wait> | Creates TCP listen port | | |
| CLOSE | <handle> | Closes TCP connection | | |
| SSTAT | | Prints status of all sockets | | |

## *SD-CARD INTERFACE*

| | | | | |
|---|---|---|---|---|
| OPEN | <handle> <filename> | Opens a file for reading and associates file handle with it | | 0xa0 |
| OPEN? | | Queries open file handles | | 0xa1 |
| NEW | <handle> <filename> | Creates file for writing. | | 0xa2 |
| APPD | <handle> <filename> | Opens existing file for writing at the end. | | 0xa3 |
| WRITE | <handle><data> | Writes data | | 0xa4 |
| READ | <handle><amount> | Reads data | | 0xa5 |
| CLOSE | <handle> | Closes opened file | | 0xa6 |
| STREAM | <handle> | Activate data streaming | | 0xa7 |
| DIR | <dir> | Shows directory | | 0xa8 |
| MKDIR | <directory> | Creates directory | | 0xa9 |
| DEL | <file> | Deletes file by name | | 0xaa |
| FSTAT? | | Shows storage card information | | 0xab |
| POS | <handle><position> | Positions file pointer | | 0xac |
| WS | <sector number> | Write sector <x> on SD Card | | 0xad |
| RS | <sector number> | Read sector <x> on SD Card | | 0xae |

## *BASIC SCRIPTING*

| | | |
|---|---|---|
| LOAD | *Basic script* | Loads BASIC script into internal memory. Finish with stop sequence. |
| LIST | *none* | Shows stored basic script |
| RUN | *none, auto, wait, autowait, manual* | Starts execution of script |

## *ADVANCED COMMANDS*

| | | | | |
|---|---|---|---|---|
| PROGFW | *none* | Burns previously loaded firmware into internal memory | | 0xd1 |
| LOADFW | | Loads firmware from disk into DataFlash. No burning process. | | 0xd0 |