TKE

TECHNISCHER KUNDENDIENST & ENTWICKLUNG
NORBERT GERHARDT

Trace Moisture Meter USB API

Firmware Revision: 2021-01-25

0. Abstract

This document formally describes the application programming interface of the TMM-1. Advanced users who wish to write their own control and data acquisition program can find here the details of the communication with the TMM-1.
The knowledge provided here is not necessary for everyday use of the TMM-1. For typical applications in the laboratory or in the industry the **Q-Moisture** software has all what's needed.

1. Syntax

In this document, syntax descriptions are written in `red`,
examples and actual commands and messages are `green`.
Brackets **[ ]** are used as placeholders. They do not appear in the protocol.

1.1. Interface

The TMM-1 features a USB serial interface device class connection through an FTDI FT240XS chip. When connected to a Windows computer, the driver will automatically be installed and a virtual COM port will be set up. Application programs can either access the driver directly (recommended) or open the COM port (less efficient). The speed is always Full Speed USB (1 MByte/s), baud rate settings of the COM port are ignored.

The drivers and documentation can be found here: https://ftdichip.com/drivers/d2xx-drivers/

There is also a Python wrapper to the driver: https://pypi.org/project/ftd2xx/

1.2. Commands

a) Simple command

```
commandname[CR]
hello[CR]
```

A simple command is the command name followed by a CR. The command will be executed immediately. Command names are not case sensitive.

b) Command with arguments

```
commandname [argument1] [argumentN][CR]
setU 25.0[CR]
password "LetMeIn"[CR]
writecal "some text" 20[CR]
```

Most commands require a list of arguments. The arguments may be of integer, float or string type, they are separated with spaces.

- Integer numbers are written in decimal ascii.
- Floating point numbers are written in decimal ascii with a decimal point. Numbers smaller than zero do not need a leading zero. Integer values can omit the trailing decimal point. Scientific notation with exponent is possible. Example: 1000 can be expressed as 1.0E+03
- Strings are surrounded by inch signs: "example string argument".

c) Command with request

```
commandname ?[CR]
backlite ?
```

Most commands accept a question mark as argument. In this case they do not carry out their usual function, but return one or more messages with information related to the command. The exact behaviour of each command is described in section 2 of this document.

1.3. Info messages

a) Simple message

```
#[ID][CR]
#0301
```

A simple message starts with a hashmark (#) and is followed by a unique identifier number. The ID has always four digits.

b) Message with arguments

```
#[ID] [Argument1] [ArgumentN][CR]
#1450 25.000
```

Similar to commands, messages may be followed by a list of arguments. They may be of integer, float or string type. Multiple arguments are separated with spaces.

c) Verbose message

```
#[ID] [Argument1] [ArgumentN] ([Explanation])[CR]
#0302 (password accepted)
#0650 "some text" 123 (configuration key and value)
```

If the verbose mode is activated, all messages are followed by an explanatory text written in parentheses. When the device is controlled by a computer, switch the verbose mode off to avoid unnecessary traffic. The verbose mode is toggled by the command *verbose*.
The first example has no argument but verbose text enabled, the second example shows a message with a string argument, an integer argument and the optional verbose text.

1.4. Error messages

Error messages are the same as Info messages but come with an exclamation mark (!) as prefix instead of the hashmark (#). The identifiers of error and info messages may overlap.
Error messages are typically returned when commands could not be executed or at any time when the device wants to report critical issues. They should be displayed to the user immediately.

Examples for error messages:

```
!2100 (filename already exists)
```

A precondition was not met, so the command could not be executed.

```
!9909 (power supply voltage too low)
```

A spontaneous fault occurred.

1.5. Command Acknowledgment

Once a command is executed, a *command done* message will be returned. This is an Info message where the last two digits of the identifier are zero. The optional verbose text contains the name of the command. Finally a prompt symbol (>) indicates that the processor is ready to accept the next command. The USB interface chip has 1kB of input buffer, so multiple commands can be sent in a row after a prompt.

```
#XX00 [(xxx command done)][CR]>
#0700 (buzzer command done)
>
```

The done message is always followed by a prompt (>). A prompt also appears when sending an empty command, i.e. a single CR byte. To establish a connection to the device, send CRs until a prompt appears. Then the device is ready to accept commands. Before the connection is established in this way, the device will not send messages by itself.

Some commands start processes that may take longer, for example a file transfer. However the command done message will be returned immediately. Other messages related to the command (for example the next chunk of file data) may appear later.

2. Commands and Messages

The following commands are accepted by the TMM microcontroller. There are some messages associated with commands, they appear in the following order:

- Immediate messages are directly returned on execution. However, it is not safe to wait for immediate messages, because not all of them appear under any condition. For example error messages come only on errors. Better use the *command done* message as acknowledgment.
- Request messages are returned when sending the command with question mark (?). The command will not be executed in this case.
- The *command done* message always appears after the command is finished.
- Further related messages may appear at any time. Immediate and request messages may also appear at any time, independent of a command.

00 - hello

Returns a greeting text and messages with the firmware date, serial number and uptime of the device.

Syntax	`hello[CR]`
Arguments	none

Immediate Messages	`#0050 "[YYYY-MM-DD]" (firmware date)`
	The firmware release date should be checked because future releases may involve changes in the communication protocol.
	
	`#0050 "[nnn]" (serial number)`
	The serial number of the device is a unique three-digit number string.
	
	`#0050 [t] (uptime in minutes)`
	The uptime in minutes is counted since the last reboot of the device.

Request Messages	none
Done Message	`#0000 (hello command done)`

01 - help

Displays a list of existing commands.

Syntax	`help[CR]`
Arguments	none
Immediate Messages	none
Request Messages	none
Done Message	`#0100 (help command done)`

02 - verbose

Switches on the explanatory text after messages. Should be switched off if the device is controlled by a computer in order to avoid excess data traffic.

Syntax	`verbose [mode][CR]`
Arguments	mode: 0 = off, 1 = on, 2 = on only for error messages (default)
Immediate Messages	none

Request Messages	`#0250 0 (verbose mode off)`
	`#0250 1 (verbose mode on)`
	`#0250 2 (verbose only for errors)`
	Returns the currently active verbose mode.

Done Message	`#0200 (verbose command done)`

03 - password

Some features are locked by default and need to be activated first.

Syntax	`password "[password]"[CR]`
Arguments	password = a string used as password

Immediate Messages	`#0301 (password denied)`
	`#0302 (password accepted)`
	The given password was correct or incorrect.
	
	`!0300 (password required)`
	The previous command could not be executed before entering the password. The only command that currently requires a password is saving factory calibration data with *save 1*.

Request Messages	none
Done Message	`#0300 (password command done)`

04 - firmware

This command is used for installing firmware updates to the microcontroller. It must be enabled first with a password. Command is not implemented.

Syntax	`firmware[CR]`
Arguments	none
Immediate Messages	none
Request Messages	none
Done Message	`#0400 (firmware command done)`

05 - reboot

Reboots the microcontroller. The reboot is carried out one second after the done message. The last cause for a reboot can be retrieved with *reboot ?*

Syntax	`reboot[CR]`
Arguments	none
Immediate Messages	none

Request Messages	`#0501 0 (power on reset)`
	This is the usual reset cause after switching the device on.
	
	`#0501 1 (brown out reset)`
	The device triggered a reset due to a supply voltage drop.
	
	`#0501 2 (watchdog reset)`
	The firmware was not executing correctly, so the watchdog timer triggered a reset after 8 seconds.
	
	`#0501 3 (software reset)`
	The user has sent a reboot command.
	
	`#0501 4 (hardware reset)`
	The reset pin at the microcontroller has been pulled low.

	`#0501 5 (config mismatch reset)`
	The microcontroller could not start due to configuration errors.
	
	`#0501 6 (overvoltage reset)`
	The microcontroller detected an overvoltage condition on its power supplies.

Done Message	`#0500 (reboot command done)`

06 - readcal

Read data from the key/value storage. This data is typically used for calibration and user configuration data. A full list of keys and possible values is to be found in chapter 4. All keys are strings.

Syntax	`readcal "[key]"[CR]`
Arguments	key = key of the key/value pair to be retrieved.
Immediate Messages	`#0650 "[key]" [value] (configuration key and value)` Returns the desired value of the given key.
Request Messages	none
Done Message	`#0600 (readcal command done)`

07 - writecal

Write data to the key/value storage. This data is used for calibration and user configuration data. A full list of keys and possible values is to be found in chapter 4. Keys are always strings, values may be of any type. Key/value pairs submitted here will be parsed and applied immediately, changed user settings will directly take effect. To store them permanently, send the *save* command.

Syntax	`writecal "[key]" [value][CR]`
Arguments	key = key of the key/value pair to be written. value = value of the key/value pair to be written.
Immediate Messages	none
Request Messages	none
Done Message	`#0700 (writecal command done)`

08 - save

The data of the key/value storage can be saved to the internal flash of the microcontroller. It will then be retrieved and applied automatically on the next powerup. There are two memory areas, one for factory calibration data and one for user settings. Saving the factory calibration requires a password. See in chapter 4, which keys go to which memory area. Do not use the save command too frequent as the flash memory tolerates only 20000 write cycles.

Syntax	`save [area][CR]`
Arguments	area: 0 = user configuration, 1 = factory calibration Selects which memory area is to be saved. If omitted, the command defaults to the user configuration area.

Immediate Messages	`#0801 (erasing flash...)`
	`#0802 (writing to flash...)`
	`#0803 (configuration successfully written to flash)`
	These messages indicate the normal sequence of storing data to the internal flash of the microcontroller.
	
	`!0800 (error while erasing flash)`
	`!0801 (error while writing to flash)`
	`!0802 (flash write buffer overflow)`
	If one of these errors occurred, the data could not be stored.

Request Messages	none
Done Message	**#0800 (save command done)**

09 - backlite

Switches the LCD backlight on or off.

Syntax	**backlite [state][CR]**
Arguments	state: 0 = off, 1 = on
Immediate Messages	none
Request Messages	**#0950 [state] (backlight state)** Reports the actual backlight state whether the light is on or off. This message also appears when the backlight is switched with the pushbutton.
Done Message	**#0900 (backlite command done)**

10 - relaya

Configures the behaviour of relay A.

Syntax	**relaya [signalnumber] [threshold][CR]**
Arguments	signalnumber = selects one of the following switching conditions: 0 Relay is always off 1 Relay is always on 2 Relay has the same state as the other relay 3 Relay has the opposite state of the other relay 4 Relay is on when the cell voltage is above the threshold 5 Relay is on when the cell voltage is below the threshold 6 Relay is on when the cell current is above the threshold 7 Relay is on when the cell current is below the threshold 8 Relay is on when the cell power dissipation is above the threshold 9 Relay is on when the cell power dissipation is below the threshold 10 Relay is on when the cell resistance is above the threshold 11 Relay is on when the cell resistance is below the threshold 12 Relay is on when the moisture value is above the threshold 13 Relay is on when the moisture value is below the threshold 14 Relay is on when the integral value is above the threshold 15 Relay is on when the integral value is below the threshold threshold = in the case of signalnumber 4...15 the switching threshold must be specified. In cases 0...3 it can be omitted or will be ignored.
Immediate Messages	none
Request Messages	**#1001 [state] (relay A state)** Informs about the actual switching state (0 = contact open or 1 = contact closed). **#1050 [signalnumber] [threshold](relay A configuration)** Reports the configuration of relay A.
Done Message	**#1000 (relaya command done)**

11 - relayb

Configures the behaviour of relay B.

Syntax	**relayb [signalnumber] [threshold][CR]**

Arguments	signalnumber = selects one of the following switching conditions: 0 Relay is always off 1 Relay is always on 2 Relay has the same state as the other relay 3 Relay has the opposite state of the other relay 4 Relay is on when the cell voltage is above the threshold 5 Relay is on when the cell voltage is below the threshold 6 Relay is on when the cell current is above the threshold 7 Relay is on when the cell current is below the threshold 8 Relay is on when the cell power dissipation is above the threshold 9 Relay is on when the cell power dissipation is below the threshold 10 Relay is on when the cell resistance is above the threshold 11 Relay is on when the cell resistance is below the threshold 12 Relay is on when the moisture value is above the threshold 13 Relay is on when the moisture value is below the threshold 14 Relay is on when the integral value is above the threshold 15 Relay is on when the integral value is below the threshold threshold = in the case of signalnumber 4...15 the switching threshold must be specified. In cases 0...3 it can be omitted or will be ignored.
Immediate Messages	none
Request Messages	**`#1101 [state](relay B state)`** Informs about the actual switching state (0 = contact open or 1 = contact closed). **`#1150 [signalnumber] [threshold](relay B configuration)`** Reports the configuration of relay B.
Done Message	**`#1100 (relayb command done)`**

12 - relayc

Switches the cell relay. When this relay is opened the negative input terminal is isolated so the zero offset can be checked. Attention: There is still voltage on the positive terminal against ground.

Syntax	**`relayc [state][CR]`**
Arguments	state: 0 = open, 1 = closed
Immediate Messages	none
Request Messages	**`#1250 [state] (relay C state)`** Informs about the actual switching state (0 = contact open or 1 = contact closed). This message also appears when the relay is switched with the pushbutton.
Done Message	**`#1200 (relayc command done)`**

13 - current

Configures the 4-20mA analogue current loop output.

Syntax	**`current [Signalnumber] [4mA-value] [20mA-value][CR]`**
Arguments	signalnumber = selects which signal to present: 0 Fixed current value in mA 1 Cell voltage in V 2 Cell current in mA 3 Power dissipation of the cell in W 4 Internal resistance of the cell in Ohm 5 Moisture value according to the conversion factor

	6 Integral value according to the integral factor 4mA-value = lower boundary of the signal range. In case of signalnumber 0 this is directly the output current. 20mA-value: upper boundary of the signal range. In case of signalnumber 0 this value can be left out or will be ignored. Both values can be interchanged for a reversed output representation.
Immediate Messages	none
Request Messages	**`#1350 [Signalnumber] [4mA-value] [20mA-value] (current loop configuration)`** Reports the active configuration of the analogue output.
Done Message	**`#1300 (current command done)`**

14 - setu

Sets the voltage of the cell voltage generator.

Syntax	**`setu [voltage][CR]`**
Arguments	voltage = 0.0 ... 25.0 V
Immediate Messages	none
Request Messages	**`#1450 [voltage] (set cell voltage)`** Reports the set cell voltage. The actual voltage at the output clamps may be lower due voltage drop at the 10 Ohm shunt resistor.
Done Message	**`#1400 (setu command done)`**

15 - seti

Sets the current limit of the cell voltage generator.

Syntax	**`seti [amps][CR]`**
Arguments	amps = 0.1 ... 100.0 mA
Immediate Messages	none

Request Messages	**`#1501 [state] (current limit state)`**
	Indicates whether the output current is limited by the set current limit. 0 = not limited, 1 = limited. When the limit is reached, the red LED lights up.
	
	`#1550 [amps] (set cell current limit in mA)`
	Reports the set current limit. The current might not reach this limit if another limiting factor such as the power limit prevents the current from going higher.

Done Message	**`#1500 (seti command done)`**

16 - setp

Sets the power limit of the cell voltage generator. Command is not implemented, the limit is fixed at 1W.

Syntax	**`setp [watts][CR]`**
Arguments	watts = 0.01 ... 1.0 W
Immediate Messages	none
Request Messages	**`#1650 (set cell power limit in Watts)`** Reports the set power limit. The current might not reach this limit if another limiting factor such as the current limit prevents the power from going higher.
Done Message	**`#1600 (setp command done)`**

17 - sett

Sets the sampling interval time in milliseconds. The sampling interval applies to the reporting of data through USB and RS232 as well as to writing on the microSD card. The 4-20 mA analog output works independent of this setting. See the *report* command for details.

Syntax	`sett [time][CR]`
Arguments	time = 10 ... 1000000 ms (i.e. 100 Hz ... 1000 s)
Immediate Messages	none
Request Messages	`#1750 [time] (sampling interval in milliseconds)` Reports the sampling interval.
Done Message	`#1700 (sett command done)`

18 - getval

Request of actual measurement values. This command works independent of the sampling interval, so values can even be received between two interval times when the sampling interval is very slow. This is useful to show a live display on a GUI.

Syntax	`getval [flags][CR]`
Arguments	flags = bitfield of flags that control the returned values: 1 = report the moisture value 2 = report the integral value 4 = report the measured cell voltage 8 = report the power supply voltage 16 = report the cell current in mA 32 = report the analogue output value (4-20mA) To receive multiple messages in a row, sum up the flags of the desired messages. Example: *getval 3* will return both messages #1801 and #1802.

Immediate Messages	`#1801 [moisture] (moisture sample value)`
	This is the moisture value which is calculated from the cell current using the conversion factor. Usually this is ppmV @ 100ml/min.
	
	`#1802 [integral] (integral sample value)`
	This is the integral of the cell current multiplied with the integral factor. Usually the value means µg water.
	
	`#1803 [volts] (actual cell voltage)`
	This is the measured cell voltage. This value is mostly a little lower than the set cell voltage due to voltage drop at the internal resistance of 10 Ohm.
	
	`#1804 [volts] (power supply voltage)`
	This is the measured power supply voltage. If both USB and the DC jack are powered, the higher of the two voltages is shown.
	
	`#1805 [amps] (actual cell current in mA)`
	This is the current through the cell. It is given in mA and is not converted to the moisture value.
	
	`#1806 [amps] (actual current loop output in mA)`
	This is the output current to be expected at the analogue 4-20mA interface.

Request Messages	none
Done Message	`#1800 (getval command done)`

19 - convunit

Defines the conversion factor and unit. The moisture value is calculated from the measured cell current in mA by multiplication with the conversion factor.

Syntax	`convunit [factor] "[unit]"[CR]`
Arguments	factor = units per mA. The conversion factor can be any 32 bit floating point value. The default value is 76.1035 which results in ppmV @ 100ml/min. unit = string to be shown on the display. Any user-defined text can be entered here. The default string is "ppmV @ 100ml/min"
Immediate Messages	none
Request Messages	`#1950 [factor] "[unit]" (conversion factor and unit)` Returns the currently used factor and unit.
Done Message	`#1900 (convunit command done)`

20 - report

Enables data reporting via USB and RS232. A message string with the measurement values will appear once per sampling interval time.

Syntax	`report [mode][CR]`
Arguments	mode: 0 = disabled, 1 = USB only, 2 = RS232 only, 3 = both
Immediate Messages	`#2001 [tc] [volts] [moisture] [integral] (ms voltage sample integral)` This message appears regularly with the time interval defined with the command *sett*. The timecode tc is given in milliseconds since the start of the reporting. It will be set back to zero when disabling the reporting. The timecode rolls over at 2^{32} ms = 1193 hours. The same string (without ID and verbose text) is put out simultaneously via RS232 if enabled.
Request Messages	`#2050 [mode] (reporting mode)` Returns the reporting mode 0 - 3.
Done Message	`#2000 (report command done)`

21 - logging

Starts and stops logging of data to a file on the microSD card. Log files are always stored in the root directory of the card. The maximum file size is 2 GB. Log file entries are created with the sampling interval set by the command *sett*. The logfile is updated once per minute, so in case of power loss, only the last minute of data will be lost.

Syntax	`logging [switch] "[filename]"[CR]`
Arguments	switch: 0 = stop logging and close file, 1 = start logging with new file filename = any string up to 31 characters, some characters are not allowed in filenames. If the filename ends with .csv, a CSV formatted file will be created. All other names result in a smaller binary file that can only be opened by the Q-Moisture software.

Immediate Messages	`!2100 (filename already exists)`
	A file with the given name already exists on the microSD card. Choose a different name.
	
	`!2101 (already logging)`
	Once the logging to a file was started, it must be stopped before logging to another file can be started.
	
	`!2102 (too little space left on sd card)`
	The memory card is full, logging can not be started.

Request Messages	`#2101 "[name]" [bytes] [ms](logfile name / size / time)` Reports the name and size in bytes of the currently open log file. Reports the

	elapsed time in ms since the start of the logging. This messages appears only if logging is active. **#2150 [state](logging state)** Reports whether logging is in progress or not. This message also appears when logging is started or stopped with the pushbutton.
Done Message	**#2100 (logging command done)**

22 - getlog

This command can either list the contents of the root directory of the microSD card or initiate a file download. It is possible to download a currently active logfile. In this way a marching waveform can be displayed while keeping a logfile copy on the microSD card even if the computer is shut down or loses connection.

Syntax	**getlog [stop] "[filename]" [start] [len][CR]**
Arguments	stop: 0 = abort the file transfer immediately (optional argument) filename = the name of the file to be transferred. start = file position where the transfer begins. The position is counted in bytes beginning with 0 at the first byte. len = number of bytes to be transferred.
Immediate Messages	**#2201 [bytes] (number of bytes of binary data following)** This message announces the transfer of a chunk of binary data. The binary block immediately follows the CR of this message. Large amounts of data are split into chunks of 512 bytes per message. **Attention**: The binary data can contain protocol symbols like # ! > CR. Make sure to skip the binary data when parsing the protocol. **#2202 (end of file reached at number of bytes)** The requested number of bytes exceeded the file size, so the file transfer terminates at the end of the file. **#2203 (file transfer terminated)** All bytes of the last transfer request have been submitted. **!2200 (busy reading file)** There is already a file transfer in progress. Wait for the end or terminate it with *getlog 0*. **!2201 (start position above file size)** The given start position is higher than the file size, therefore no data could be transferred.
Request Messages	**#2210 [state] (sd card presence)** Informs about the presence of a microSD card. 0 = no card inserted, 1 = card present. This message appears whenever a card is inserted or removed or if the getlog request command could not be executed due to missing card. **#2251 "[name]" [size] (filename and size in bytes)** This message appears once per file in the root directory. **#2252 (directory is empty)** The root directory is empty, so no files can be listed.
Done Message	**#2200 (getlog command done)**

23 - delete

Deletes a file on the microSD card.

Syntax	**delete "[filename]"[CR]**

Arguments	filename = file to be deleted, case sensitive
Immediate Messages	`!2300 (file in use)` The file can not be deleted because it is opened either as log file or in a file transfer.
Request Messages	none
Done Message	`#2300 (delete command done)`

24 - integral

Start/Stop the integration of the cell current to determine the amount of water.

Syntax	`integral [switch][CR]`
Arguments	switch: 0 = stop integration and keep the last value, 1 = reset the value to zero and start integrating
Immediate Messages	none
Request Messages	`#2450 [state](integrating state)` Reports whether an integration is running or not. This message also appears when the integration ist started or stopped from the pushbutton.
Done Message	`#2400 (integral command done)`

25 - intunit

Defines the integral factor and unit. The amount of water is calculated from the measured electrical charge in mAs (milliampere-seconds) by multiplication with the integral factor.

Syntax	`intunit [factor] "[unit]"[CR]`
Arguments	factor = units per mAs. The integral factor can be any 32 bit floating point value. The default value is 0.09383 which results in µg water. unit = string to be shown on the display. Any user-defined text can be entered here. The default string is "~g Water". Notice: The tilde sign will be shown as µ on the display.
Immediate Messages	none
Request Messages	`#2550 [factor] "[unit]" (integral factor and unit)` Returns the currently used factor and unit.
Done Message	`#2500 (intunit command done)`

26 - format

Formats the microSD card with FAT32. All data will be lost.

Syntax	`format[CR]`
Arguments	none
Immediate Messages	`!2600 (can not format while files are open)` Close a log file and terminate a file transfer before formatting.
Request Messages	none
Done Message	`#2600 (format command done)`

3. System Error Messages

These messages are not related to specific commands and can show up at any time.

`!9900 (command unknown)`
A command with the given name does not exist.

`!9901 (command syntax error)`
The argument list given with the command has a syntax error.

`!9902 (input buffer overflow)`
Too many characters have been received, but no CR.

`!9903 (argument out of range)`
The command could not be executed because an argument exceeded the possible range.

`!9904 (wrong number of arguments)`
The controller expects a different number of arguments for the last command.

`!9905 (string too long)`
The argument list contains a string with more than 31 characters.

`!9906 (unknown key)`
The command *readcal* or *writecal* was sent with a key that is not in the list of possible keys (see chapter 4), or during startup a key has been found in the key/value storage that is not the list of possible keys.

`!9907 (nothing to request)`
The command was given with a question mark, but this command does not support requests.

`!9908 (string contains forbidden characters)`
Characters that are used for the protocol are not allowed within strings. These are: # ! > \0

`!9909 (power supply voltage too low)`
Some operations can only be carried out with sufficient power reserve in the buffer capacitor.

`!9910 (calibration value missing)`
During startup the device could not be initialised with all necessary calibration values. Default values will be used instead. The measuring results will not be accurate.

`!9911 (not enough RAM)`
The allocation of work memory for some internal operation failed.

```
!9920 0 (no sd card inserted)
!9920 1 (disk hardware io error)
!9920 2 (internal filesystem error)
!9920 3 (drive not ready)
!9920 4 (file not found)
!9920 5 (path not found)
!9920 6 (invalid path name)
!9920 7 (file access denied)
!9920 8 (file access prohibited)
!9920 9 (file or directory object invalid)
!9920 10 (drive is write protected)
!9920 11 (logical drive number invalid)
!9920 12 (file system not enabled)
!9920 13 (no filesystem volume found)
!9920 14 (format aborted due to parameter error)
!9920 15 (filesystem access timeout)
!9920 16 (file or path locked by other task)
!9920 17 (could not allocate filesystem buffer)
!9920 18 (too many open files)
```

```
!9920 19 (filesystem parameter invalid)
!9920 20 (not enough free volume)
!9920 21 (filesystem type not supported)
!9920 22 (unexpected filesystem type)
!9920 23 (function not supported in filesystem)
!9920 24 (unknown filesystem error)
```
These are file system related error messages. The above errors will show up whenever the microcontroller fails to carry out an operation on the microSD card.

4. Key / Value Storage

The key/value storage holds the factory calibration and user configuration data. Key/value pairs are written with the *writecal* command and can be read by the *readcal* command. To save the data permanently to the internal flash of the microcontroller send the *save* command. Saved data will be applied on the next powerup. Keys are case sensitive.

4.1 Factory calibration data

You can write these key/values with *writecal* and they become effective immediately. To save them permanently, a password must be entered first using the *password* command.

Key	Value	Meaning
serial_number	"[xxx]"	A unique string identifying this device.
hw_revision	0 = K-BOX 1 = TMM 1.0 2 = TMM 1.2	Hardware Variant
cal_i_gain	0.5 ... 1.5	Gain correction of the 24 bit cell current ADC
cal_volt_5V	0 ... 1023	5V calibration point for the cell voltage ADC
cal_volt_20V	0 ... 1023	20V calibration point for the cell voltage ADC
cal_umax_5V	0 ... 4095	5V calibration point for the generator voltage DAC
cal_umax_20V	0 ... 4095	20V calibration point for the generator voltage DAC
cal_imax_1mA	0 ... 4095	1mA calibration point for the generator current limit DAC
cal_imax_100mA	0 ... 4095	100mA calibration point for the generator current limit DAC
cal_curr_4mA	0 ... 16383	4mA analogue output DAC calibration
cal_curr_20mA	0 ... 16383	20mA analogue output DAC calibration

4.2 User configuration data

Most but not all key/values take effect immediately and are identical to the related command.

Key	**Value**	**Meaning**
disp_rotation	0 = normal, 1 = upside down	Display rotation. Takes effect on next reboot.
conv_unit	[factor] "[unit]"	Conversion factor and unit. Same as command *convunit*.
integral_unit	[factor] "[unit]"	Integral factor and unit. Same as command *intunit*.
def_u	0.0 ... 25.0 V	Default generator voltage. Same as command *setu*.
def_i	0.1 ... 100.0 mA	Default generator current limit. Same as command *seti*.
def_p	0.1 ... 1.0 W	Default generator power limit. Same as command *setp*.
def_t	10 ... 1000000 ms	Default sampling interval time Same as command *sett*.
def_backlite	0 = off, 1 = on	Default Backlight state. Same as command *backlite*.
def_relayA	[signalnumber] [threshold]	Default relay A configuration. Same as command *relaya*.
def_relayB	[signalnumber] [threshold]	Default relay B configuration. Same as command *relayb*.
def_relayC	0 = open, 1 = closed	Default relay C state. Same as command *relayc*.
def_current	[signalnumber] [4mA] [20mA]	Default analogue output configuration. Same as command *current*.
def_filename	"[filename]"	Default log file name. This filename is used when starting the logging with the pushbutton or when autostart logging on powerup is active. The name will be enhanced with a number suffix if the file already exists. filename.csv becomes filename_1.csv and so on.
def_logging	0 = do nothing, 1 = start logging	Autostart logging on powerup. If set the device automatically creates a new log file with the default filename when powered up.
def_reporting	[mode]	Autostart reporting on powerup. If set the device starts to send measurement reports to USB and/or RS232 right after power up. On USB the reports are first visible if the receiving computer has established the connection by sending CRs (see chapter 1.5).