PalmSens
Compact Electrochemical Interfaces

EmStat4 communication protocol V1.0

Rev. 2021-10-001

Contents

1 Introduction

This document describes the “online” communication protocol of the EmStat4.
Initial communication with an EmStat4 is always done using this online communication. Measurements can be started by sending a MethodSCRIPT, more information about MethodSCRIPT can be found here:

http://www.palmsens.com/methodscript

Terminology

PGStat:	Potentiostat / Galvanostat
CE:	Counter Electrode
RE:	Reference Electrode
WE:	Working Electrode
Technique:	A standard electrochemical technique
Iteration:	A single execution of a loop
RAM:	Random-Access (volatile) Memory
Flash:	Flash (non-volatile) memory
SD-card:	Internal or external flash memory card
MethodSCRIPT:	PalmSens human readable measurement script format

2 Communication

2.1 UART settings

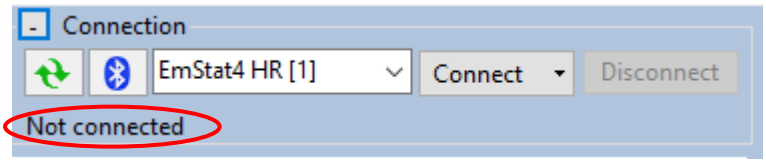
The EmStat4 communicates using 3.3V UART (Serial Port) with the following settings:

Setting	Value	Description
Signal level	3.3V	
Baudrate	230400	Baud (bps)
Number of data bits	8	
Number of stop bits	1	
Parity	None	
Handshaking	None or RTS/CTS	No hardware handshaking is used when RTS/CTS pins are unconnected.

2.2 Connection viewer

PSTrace version 5.6 or higher has a hidden feature, which is useful when the communication protocol is used for development of software for EmStat4.

PSTrace will open the 'Connection viewer' window when you double click the "Not connected" label before connecting to the device.



Double click in this area before connecting to open the Connection viewer

The connection viewer window. All information in red is sent from the PC to the device and the green information is sent by device to the device.

2.3 Communication protocol

All commands and responses are terminated with a newline character ('\n' or 0x0A). Commands will echo the first character of the command and then respond with command specific data. When the command has finished executing a newline character is returned. If an error occurs during the execution of a command, the error is returned just before the final newline. See section "Error codes" for more information about errors.

2.4 Communication modes

The device can be in two communication modes:

1. Idle mode
2. Script execution mode

Idle commands cannot be sent during the execution of a MethodSCRIPT. Script execution commands can only be sent when the device is executing a MethodSCRIPT.

2.5 Commands overview

Idle commands

Command	Description	See section
't'	Get the device firmware version	3.1
'S'	Set register value	3.2
'G'	Get register value	3.3
'l'	Load (parse) MethodSCRIPT to RAM	3.4
'r'	Run (execute) loaded MethodSCRIPT	3.5
'e'	Load and execute MethodSCRIPT (same as 'l' followed by 'r')	3.6
"Fmscr"	Store loaded MethodSCRIPT to non-volatile (Flash) memory	3.7
"Lmscr"	Retrieve MethodSCRIPT from non-volatile memory to RAM	3.8
'i'	Get device serial number	3.9
'v'	Get MethodSCRIPT version	3.10
"dlfw"	Enter bootloader mode.	3.11
"fs_mount"	Mount the file storage flash memory	3.12.1
"fs_dir"	Print the content of the directory	3.12.2
"fs_get"	Print the content of a file	3.12.3
"fs_del"	Remove a file or directory (recursively)	3.12.4
"fs_info"	Display SD-card information (free/used/total space)	3.12.5
"fs_format"	Format the SD-card (fs_clear is preferred above fs_format)	3.12.6
"fs_clear"	Removes all files and folders from the SD-card	3.12.7
"fs_put"	Write ASCII data to file	3.12.8

Table 1; Idle commands

Script execution commands

Command	Description	See section
'h'	Hold execution of the running MethodSCRIPT	4.1
'H'	Resume execution of the halted MethodSCRIPT	4.2
'Z'	Abort execution of the running MethodSCRIPT	4.3
'Y'	Abort current measurement loop	4.4

Table 2; Script execution commands

3 Idle commands

3.1 't' command

Sending "t\n" to the device returns the firmware version of the device.

Note: unlike other commands this command responses with multiple newline ("\n") separated strings terminated by a "*\n"

Response
tes4_?r#mmm dd yyyy hh:mm:ss\n
R*\n

t	Echo of command
es4_?r	EmStat4 identifier ("es4_lr" or "es4_hr")
xxxx	represents the firmware version without the decimal point, e.g. when xxxx = 1000, the firmware version is 1.0.00
#	separator
mmm	month in short-form e.g. "Jun"
dd	2 digit day number
yyyy	4 digit year number
hh	2 digit hour number
mm	2 digit minutes number
ss	2 digit seconds number
R	Release firmware
*\n	End of version command

Example
t\n

Example response
tes4_lr1000#Jun 7 2021 16:51:38\n
R*\n

3.2 'S' command

Sending "Sxxyy..yy\n" to the device sets register xx to value yy..yy.

Response
S\n

Where:

xx	2 digit hexadecimal register number
yy..yy	Hexadecimal digits representing the value of the register to be set. The number of digits depends on the register number.

See section "Registers" for detailed information.

3.3 'G' command

Sending "Gxx\n" to the device gets the value of register xx.

Response

Gyy.yy\n	;Register 08 has value yy..yy

Where:

xx	2 digit hexadecimal register number
yy..yy	Hexadecimal digits representing the value of the register. The number of digits depends on the register number.

See section "Registers" for detailed information.

Example:

G08\n	;Get the value of register 08 (autorun enabled)

3.4 'l' command

Sending "l\n" to the device loads and parses the MethodSCRIPT following the command. The end of the script is indicated by an empty line containing only a "\n" character. If the status of the loading indicates no errors, the script can be executed by the 'r' command (see section 3.5).

Response
l\n

Example
l\n
send_string "hello world"\n
\n

3.5 'r' command

Sending "r\n" to the device executes a loaded MethodSCRIPT.

Response
r\n

3.6 'e' command

Sending "e\n" to the device loads and parses the MethodSCRIPT following the command, and then executes the script if no errors are returned.

Response
e\n
… Script output
\n

Example
e\n
send_string "hello world"\n
\n

Example response
e\n
Thello world\n
\n

3.7 'Fmscr' command

Sending "Fmscr\n" to the device stores a loaded MethodSCRIPT to non-volatile memory.

Response
F\n

3.8 'Lmscr' command

Sending "Lmscr\n" to the device loads a stored MethodSCRIPT from non-volatile memory. It can now be started with the 'r' command. Note that it is only possible to load scripts with a MethodSCRIPT version that matches the version used by the firmware.

Response
L\n

3.9 'i' command

Sending "i\n" to the device gets the device serial number.

Response
iXXXXXXXXXXX\n

Example
i\n

Where the first character is the echo of the command character 'i' followed by the 12-character serial number.

Example response
iES4LR20B0005\n

3.10 'v' command

Sending "v\n" to the device prints the MethodSCRIPT version.

Response
vXXXX\n

Where the first character is the echo of the command character 'v' followed by the 4-digit hexadecimal version number.

Example
v\n

Example response
v0003\n

3.11 'dlfw' command

Sending "dlfw\n" to the device resets the device in bootloader mode.

Response
d\n

3.12 Filebrowser commands

The EmStat4 can read and write files from/to the embedded file storage flash-memory. The filebrowser interface is provided to interact with this storage medium and supports data in ASCII format.

3.12.1 'fs_mount' command – mount the file storage flash

This command mounts the file storage flash.

Response
f\n

Example
fs_mount\n

3.12.2 'fs_dir' command – show directory

The command "fs_dir PATH\n" prints all names of files and directories in the directory indicated by the parameter PATH. The EmStat4 will respond with an "f\n" followed by the lines containing the files/directories. Directories are not listed separately on the EmStat4.

Response

f\n
YYYY-MM-DD HH:mm:SS;TYP;SIZE;NAME\n
…
YYYY-MM-DD HH:mm:SS;TYP;SIZE;NAME\n
\n

Where:

YYYY	Year
MM	Month (1-12)
DD	Day (1-31)
HH	Hour (0-23)
mm	Minute (0-59)
SS	Second (0-59)
TYP	Type of file, always "FIL"
SIZE	File size in bytes
NAME	Full path of file

Example

fs_dir /measurements\n

Prints the names of the files/folders in the /measurements directory.

Example response

f\n
2019-12-31 11:34:18;FIL;100;log.txt\n
2019-12-31 11:34:23;FIL;101;info.txt\n
2019-12-31 11:34:27;FIL;800;error_codes.csv\n
\n

3.12.3 'fs_get' command – get file content

The command 'fs_get PATH\n' prints the contents of the requested file. The end of the file is indicated with a file separator character (0x1C).

Note: the filebrowser does not support the transmission of binary files.
Note2: when UART with no hardware handshaking (RTS/CTS) is used the EmStat4 transmits the data as fast as it can and will not wait for the host-system.

Response
f\n
DATA
FS\n

Where:
DATA File data
FS File separator character (0x1C) indicating end of file

Example
fs_get /measurements/my_lsv_file.data\n

Returns the content of the file "/measurements/my_lsv_file.data".

Example response
f\n
v0003\n
Pda7F9E6A6u;ba51FC060p,10,207\n
Pda7FB6CFCu;ba5C994C0p,10,207\n
Pda7FCF353u;ba6731714p,10,207\n
Pda20B3D38n;ba71CD01Bp,10,207\n
Pda8000000 ;ba7C6A479p,10,207\n
\n

3.12.4 'fs_del' command – remove file/directory

The command 'fs_del PATH\n' removes the file or directory (recursively) specified by PATH.

Response
f\n

Example
fs_del /log.txt\n

Removes the file "/log.txt".

3.12.5 'fs_info' command – get information from file storage flash

The command "fs_info\n" returns the current used space, free space and file storage flash size.

Example
fs_info\n

Response
f\n
used:XX..XXkB free: XX..XXkB total: XX..XXkB\n

Where:
XX..XX Number of kilobytes as a decimal number, length is variable.

3.12.6 'fs_format' command – Format file storage flash

This command formats the file storage flash with the filesystem. As a side-effect all content of the file storage flash is removed.

Note: This is not the preferred way to clear the file storage flash. For that use the 'fs_clear' command.
Note2: The formatting procedure can take some time. The device will return "f" to indicate the start of the format and a "\n" when the formatting was successful or an error code if not.

Response
f\n

Example
fs_format\n

3.12.7 'fs_clear' command – remove all files and directories

This command removes all files and directories on the file storage flash.

Response
f\n

Example
fs_clear\n

3.12.8 'fs_put' command – write ascii data to file

This command writes arbitrary ascii data to a file

Response
f\n

Example

fs_puts /filefolder/filename.txt\n	;open the file '/filefolder/filename.txt' for writing
0123456789ABCDEF	;data '0123456789ABCDEF' is written
\x1C\n	;Write FS (0x1C) and \n to close file to finish command

4 Script execution commands

To control the flow of execution of a running MethodSCRIPT, these commands can abort, pause and resume the execution of the script or skip the current command.

4.1 'h' command

Sending "h\n" to the device holds a running MethodSCRIPT. End of line \n is omitted in this example.

Example:

```
e
var c
var p
set_pgstat_mode 2
set_cr 100m
cell_on
meas_loop_lsv p c -1 1 10m 1
  pck_start
  pck_add p    <- sending "h\n" will hold the script at the next command
  pck_add c
  pck_end
endloop
on_finished:
cell_off
```

4.2 'H' command

Sending "H\n" to the device resumes a halted MethodSCRIPT. End of line \n is omitted in this example.

Example:

```
e
var c
var p
set_pgstat_mode 2
set_cr 100m
cell_on
meas_loop_lsv p c -1 1 10m 1
  pck_start
  pck_add p        <- sending "H\n" will resume the halted script
  pck_add c
  pck_end
endloop
on_finished:
cell_off
```

4.3 'Z' command

Sending "Z\n" to the device aborts a running MethodSCRIPT. The current iteration of any measurement loop will be completed, then the script execution will jump to the "on_finished:" tag. End of line \n is omitted in this example.

Example:

```
e
var c
var p
set_pgstat_mode 2
set_cr 100m
cell_on
meas_loop_lsv p c -1 1 10m 1
  pck_start
  pck_add p
  pck_add c    <-  sending "Z\n" within the loop will abort the script
                   and jump to the "on_finished:" tag.
  pck_end
endloop
on_finished:
cell_off
```

4.4 'Y' command

Sending "Y\n" to the device breaks the execution of the current MethodSCRIPT loop after the current iteration of the loop has finished. End of line \n is omitted in this example.

Example:

```
e
var c
var p
set_pgstat_mode 2
set_cr 100m
cell_on
meas_loop_lsv p c -1 1 10m 1
  pck_start
  pck_add p
  pck_add c    <- sending "Y\n" within the loop will abort the
                  loop after finishing the current iteration
  pck_end
endloop
on_finished:
cell_off
```

5 Registers

The internal registers are used to set and get local settings like serial-number or autorun enable/disable.

Value	access mode	Description	See section
0x04	Read only	License register	5.1
0x05	Read only	UID	5.2
0x06	Read only	Device serial	5.3
0x09	Read / Write	Advanced options	5.4
0x81	Write only	NVM commit	5.5
0x85	Write only	Factory defaults restore	5.6

5.1 License register

Contains the licenses programmed into this EmStat4. For more information contact PalmSens.

Read response
Gxxxxxxxxxxxxxxxx\n

Where:

G	echo of 'G' command
xxxxxxxxxxxxxxxx	16 hexadecimal digit license code
\n	End of command

Example
"G04\n" gets the license register.

5.2 UID

Contains the 96bits Unique Identifier based on the CPU_ID

Read response
"Gxxxxxxxxxxxxxxxxxxxxxxxx\n"

Where:

G	echo of 'G' command
xxxxxxxxxxxxxxxxxxxxxxxx	24 hexadecimal digit UID
\n	End of command

Example
"G05\n" gets the UID

5.3 Device serial

Contains the device serial number.

Read response
"Gttyybbbbnnnnnnnn\n"

Where:

G	echo of 'G' command
tt	Device type, hexadecimal representation
yy	Year, hexadecimal representation
bbbb	Batch nr, hexadecimal representation.

nnnnnnnn	Device ID, hexadecimal representation. Unique within all devices of the same type, year and batch.
\n	End of command

Example
"G06\n" gets the serial number of the device.

5.4 Advanced options

Contains the advanced option setting bitmask. Generic options are stored from the MSB-side while device specific options start at the LSB side.

In addition to the device specific bits there is currently one generic option bit for "CRC16 protocol extension". This switches the device communicate in the CRC16-protocol format (see chapter "CRC16 protocol extension").To enable it write "80000000" to the option bits. It can be disabled with "00000000".

To enable multiple options their bitmasks should be combined with a "bitwise or" operation. For example, "80000001" enables both extended voltage range and the crc16 protocol extension. Note however that writing new values overwrite all previous bits. So "80000000" also disables the extended voltage range.

Read response
G09XXXXXXXX\n

Where:

XXXXXXXX	8 digit hexadecimal representing contents of settings register

Write Response
S\n

Example
"S0980000000\n" will enable the CRC16 mode.
"S0900000000AA9D43\n" will disable the CRC16 mode. Includes the CRC required in this mode.

5.5 NVM commit

Writing "1234ABCD" (hexadecimal) to the NVM register commits the changes to the Non Volatile Memory. This makes any changes to NVM settings persistent after reboot.

Write response
S\n

Example
"S811234ABCD\n" will commit the changes to the NVM.

5.6 Factory defaults restore

Writing "DAEF2531" to the register 0x85 register restores the factory defaults including factory calibration.

Write response
S\n

Example
"S85DAEF2531\n" will restore the factory defaults

6 CRC16 protocol extension

6.1 Introduction

For certain applications of the EmStat4, data validity is of critical importance. For this purpose all data communication from and to the EmStat has to be verifiable. Since UART is the underlying protocol of all communication with the device it is possible that bits get flipped or entire bytes are missed, compromising integrity. This chapter describes an extension that adds CRC and sequence-ID fields to the EmStat4 communication protocol to allow for verification of the received data. The CRC-extension will be selectable in the EmStat4's flash configuration by setting the corresponding option bit (by issuing the command "S0980000000" in normal mode).

6.2 Protocol extension

The CRC-extension adds a sequence ID field and CRC-16 field to each line before the newline separator (\n). The sequence ID field allows the receiver to detect if there are missing lines. It consists of 2 hexadecimal characters that are incremented after each line and rolls over after 255 (0xFF). At start up both EmStat4 and host start at 0x00. The sequence IDs of both devices are incremented independently. The CRC-16 field makes it possible to verify the content of each line. It is appended after the sequence ID field and printed in a 4 digit hexadecimal format. The CRC-16-CCITT polynomial 0x1021 with initial value 0xFFFF are used to calculate the CRC over the entire string (including the sequence ID field and excluding the newline).
To give the host more security that the data is actually received by the EmStat4, the EmStat4 will acknowledge every received line with an acknowledge in the format "<xx>" (without quotes) followed by the regular header, where xx is the hexadecimal value of the sequence to acknowledge. The host should not acknowledge received data since the EmStat4 does not expect this.

The EmStat4 will respond mostly in the same way as it does without CRC-extension. An exception is with MethodSCRIPT related commands ('e' and 'l'). These will normally return with just a letter without newline and a send the newline when the entire script is received. Since this would interfere with the acknowledge messages it was decided that when the CRC16-extension is enabled it will add an additional newline directly after the command response letter.

The line format for communication will be:

[Line] [Sequence ID] [CRC-16] [\n]

Line format for acknowledge:

[<] [Sequence to acknowledge] [>] [Sequence ID] [CRC-16] [\n]

To notify the host about any detected error during communication the following error codes are added:

Name	Value	Description
STATUS_COMM_CRC_ERR	0x2B	CRC of received line was incorrect
STATUS_COMM_SEQUENCE_WARN	0x2C	ID of received line was not the expected value
STATUS_COMM_LENGTH_ERR	0x2D	Received line was too short to extract a header

6.3 Examples

Example command without CRC-extension:

t\n	From host
tes4_lr1000#Jun 7 2021 16:51:38\n	From EmStat4
R*\n	From EmStat4

Example command with CRC-extension:

t0A9524\n	From host
<0A>04935F\n	From EmStat4
tes4_hr1000#Jun 7 2021 16:51:380569DC\n	From EmStat4
D*47EE4F\n	From EmStat4

Note: "\n" is the newline character, initial sequence IDs are 0x0A for the host and 0x45 for the EmStat4.

MethodSCRIPT example command without CRC-extension:

e\n	From host
e	From EmStat4 (note: no \n)
send_string "Hello World!"\n	From host
\n	From host
THello World!\n	From EmStat4
\n	From EmStat4 (\n to close command)

MethodSCRIPT example command with CRC-extension:

e03BFA2\n	From Host
<03>4CFEF6\n	From EmStat4
e4D7D16\n	From EmStat4
send_string "Hello World!"04640F\n	From Host
<04>4ECF1D\n	From EmStat4
057E6C\n	From Host
<05>4F89CA\n	From EmStat4
50D13C\n	From EmStat4
THello World!51D393\n	From EmStat4
52F17E\n	From EmStat4

7 Error codes

After sending a command to the device, the device may respond with an error.
When loading or executing MethodSCRIPT the device may respond with specific MethodSCRIPT errors described in "MethodSCRIPT Vx_x.pdf".

Online communication error format:

c!XXXX\n

Where:

c = Echo of the first character of the command
XXXX = The error code, see “Table 3; Error codes”

Code (Hex)	Name	Description
0001	STATUS_ERR	An unspecified error has occurred
0002	STATUS_INVALID_VT	An invalid Value Type has been used
0003	STATUS_UNKNOWN_CMD	The command was not recognized
0004	STATUS_REG_UNKNOWN	Unknown Register
0005	STATUS_REG_READ_ONLY	Register is read-only
0006	STATUS_WRONG_COMM_MODE	Communication mode invalid
0007	STATUS_BAD_ARG	An argument has an unexpected value
0008	STATUS_CMD_BUFF_OVERFLOW	Command exceeds maximum length
0009	STATUS_CMD_TIMEOUT	The command has timed out
000A	STATUS_REF_ARG_OUT_OF_RANGE	A var has a wrong identifier
000B	STATUS_OUT_OF_VAR_MEM	Cannot reserve the memory needed for this var
000C	STATUS_NO_SCRIPT_LOADED	Cannot run a script without loading one first
000D	STATUS_INVALID_TIME	The given (or calculated) time value is invalid for this command
000E	STATUS_OVERFLOW	An overflow has occurred while averaging a measured value
000F	STATUS_INVALID_POTENTIAL	The given potential is not valid
0010	STATUS_INVALID_BITVAL	A variable has become either “NaN” or “inf”
0011	STATUS_INVALID_FREQUENCY	The input frequency is invalid
0012	STATUS_INVALID_AMPLITUDE	The input amplitude is invalid
0013	STATUS_NVM_ADDR_OUT_OF_RANGE	Non-volatile Memory address invalid
0014	STATUS_OCP_CELL_ON_NOT_ALLOWED	Cannot perform OCP measurement when cell on
0015	STATUS_INVALID_CRC	CRC invalid
0016	STATUS_FLASH_ERROR	An error has occurred while reading / writing flash
0017	STATUS_INVALID_FLASH_ADDR	An error has occurred while reading / writing flash
0018	STATUS_SETTINGS_CORRUPT	The device settings have been corrupted
0019	STATUS_AUTH_ERR	Authentication error
001A	STATUS_CALIBRATION_INVALID	Calibration invalid
001B	STATUS_NOT_SUPPORTED	This command or part of this command is not supported by the current device
001C	STATUS_NEGATIVE_ESTEP	Step Potential cannot be negative for this technique
001D	STATUS_NEGATIVE_EPULSE	Pulse Potential cannot be negative for this technique
001E	STATUS_NEGATIVE_EAMP	Amplitude cannot be negative for this technique
001F	STATUS_TECH_NOT_LICENCED	Product is not licenced for this technique
0020	STATUS_MULTIPLE_HS	Cannot have more than one high speed and/or max range mode enabled (EmStat4)
0021	STATUS_UNKNOWN_PGS_MODE	The specified PGStat mode is not supported
0022	STATUS_CHANNEL_NOT_POLY_WE	Channel set to be used as Poly WE is not configured as Poly WE
0023	STATUS_INVALID_FOR_PGSTAT_MODE	Command is invalid for the selected PGStat mode

0024	STATUS_TOO_MANY_EXTRA_VARS	The maximum number of vars to measure has been exceeded
0025	STATUS_UNKNOWN_PAD_MODE	The specified PAD mode is unknown
0026	STATUS_FILE_ERR	An error has occurred during a file operation
0027	STATUS_FILE_EXISTS	Cannot open file, a file with this name already exists
0028	STATUS_ZERO_DIV	Variable devided by zero
0029	STATUS_UNKNOWN_GPIO_CFG	GPIO pin mode is not known by the device
002A	STATUS_WRONG_GPIO_CFG	GPIO configuration is incompatible with the selected operation
002B	STATUS_COMM_CRC_ERR	CRC of received line was incorrect (CRC16-ext)
002C	STATUS_COMM_SEQUENCE_WARN	ID of received line was not the expected value (CRC16-ext)
002D	STATUS_COMM_LENGTH_ERR	Received line was too short to extract a header (CRC16-ext)
002E	STATUS_SETTINGS_NOT_INITED	Settings are not initialized
002F	STATUS_INVALID_CHAN	Channel is not available for this device
0030	STATUS_CAL_ERROR	Calibration process has failed
0031	STATUS_COMM_DISCONNECT	Comm interface disconnected during ongoing communication
0032	STATUS_CELL_OVERLOAD	Critical cell overload, aborting measurement to prevent damage.
0033	STATUS_FLASH_ECC_ERR	Flash ECC error has occurred
0034	STATUS_FLASH_PROGRAM_FAIL	Flash program operation failed
0035	STATUS_FLASH_ERASE_FAIL	Flash Erase operation failed
0036	STATUS_FLASH_LOCKED	Flash page/block is locked
0037	STATUS_FLASH_WRITE_PROTECTED	Flash write operation on protected memory
0038	STATUS_FLASH_BUSY	Flash is busy executing last command.
0039	STATUS_FLASH_BAD_BLOCK	Operation failed because block was marked as bad
003A	STATUS_FLASH_INVALID_ADDR	The specified address is not valid
003B	STATUS_FS_MOUNT_ERR	An error has occurred while attempting to mount the filesystem
003C	STATUS_FS_FORMAT_ERR	An error has occurred while attempting to format the filesystem memory
003D	STATUS_SPI_TIMEOUT	A timeout has occurred during SPI communication
003E	STATUS_TIMEOUT	A timeout has occurred
003F	STATUS_CALIBRATIONS_LOCKED	The calibration registers are locked
0040	STATUS_FLASH_NOT_SUPPORTED	Memory module not supported.
0041	STATUS_FS_INVALID_FORMAT	Flash memory format not recognized or supported
0042	STATUS_REGISTER_ACCESS_DENIED	Access to the register is denied
0043	STATUS_REG_WRITE_ONLY	Register is write-only
4000	STATUS_SCRIPT_SYNTAX_ERR	The script contains a syntax error
4001	STATUS_SCRIPT_UNKNOWN_CMD	The script command is unknown
4002	STATUS_SCRIPT_BAD_ARG	An argument was invalid for this command
4003	STATUS_SCRIPT_ARG_OUT_OF_RANGE	An argument was out of range
4004	STATUS_SCRIPT_UNEXPECTED_CHAR	An unexpected character was encountered
4005	STATUS_SCRIPT_OUT_OF_CMD_MEM	The script is too large for the internal script memory
4006	STATUS_SCRIPT_UNKNOWN_VAR_TYPE	The variable type specified is unknown

4007	STATUS_SCRIPT_VAR_UNDEFINED	The variable has not been declared
4008	STATUS_SCRIPT_INVALID_OPT_ARG	This optional argument is not valid for this command
4009	STATUS_SCRIPT_INVALID_VERSION	The stored script is generated for an older firmware version and cannot be run
400A	STATUS_SCRIPT_INVALID_DATATYPE	The parameter datatype (float/int) is not valid for this command
400B	STATUS_SCRIPT_NESTED_MEAS_LOOP	Measurement loops cannot be placed inside other measurement loops
400C	STATUS_SCRIPT_UNEXPECTED_CMD	Command not supported in current situation
400D	STATUS_SCRIPT_MAX_SCOPE_DEPTH	Scope depth too large
400E	STATUS_SCRIPT_INVALID_SCOPE	The command had an invalid effect on scope depth. (for example "if" directly followed by an "endif" statement)
400F	STATUS_SCRIPT_INDEX_OUT_OF_RANGE	Array index out of bounds
4010	STATUS_SCRIPT_I2C_NOT_CONFIGURED	I2C interface was not initialized
4011	STATUS_SCRIPT_I2C_UNHANDLED_NACK	NAck flag not handled by script
4012	STATUS_SCRIPT_I2C_ERR	Something unexpected went wrong with I2C.
4013	STATUS_SCRIPT_I2C_INVALID_CLOCK	I2C clock frequency not supported by hardware
4014	STATUS_SCRIPT_HEX_OR_BIN_FLT	Non integer SI vars cannot be parsed from hex or binary representation
4015	STATUS_INVALID_WAKEUP_SOURCE	The selected (combination of) wake-up source is invalid
4016	STATUS_WAKEUP_TIME_INVALID	RTC was selected as wake-up source with invalid time argument
4017	STATUS_SCRIPT_ARRAYSIZE_MISMATCH	Array size does not match expected size
4018	STATUS_SCRIPT_UNEXPECED_END	The script has ended unexpectedly.
4019	STATUS_SCRIPT_DEVICE_NOT_MULTI	The script command is only valid for a multichannel (combined) device
4020	STATUS_SCRIPT_TIMEOUT	A timeout has occurred for one of the script commands
7FFF	STATUS_FATAL_ERROR	A fatal error has occurred, the device must be reset
8000	STATUS_DEVICE_SPECIFIC	Device specific error occurred
8001	STATUS_DS_SELFTEST_CRYSTAL	Switching to 16 MHz crystal failed

Table 3; Error codes

8 Version changes

Version 1.0

- Initial version
- Compatible with firmware version: V1.0.0