PalmSens
Compact Electrochemical Interfaces

EmStat Pico communication protocol V1.3

Rev. 01

Contents

1 Introduction

This document describes the "online" communication protocol of the EmStat Pico.
Initial communication with an EmStat Pico is always done using this online communication.
Measurements and other scripts 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
NVM:	Non-Volatile Memory, memory that retains data after reset
Flash:	A specific type of Non-Volatile Memory
Storage medium:	Internal or external file storage medium (SD-card or similar)
MethodSCRIPT:	PalmSens human readable measurement script format

2 Communication

2.1 UART settings

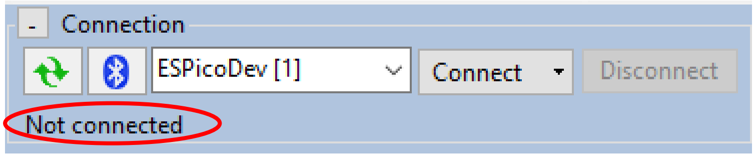
The EmStat Pico communicates using 3.3V UART (Serial Port) with the following settings:

Setting	Value	Description
Signal level	3.3V	
Baudrate	230400	Baud (bps)
Number of data bits	8	
Number of stop bits	1	
Parity	None	
Handshaking	No	No handshaking used

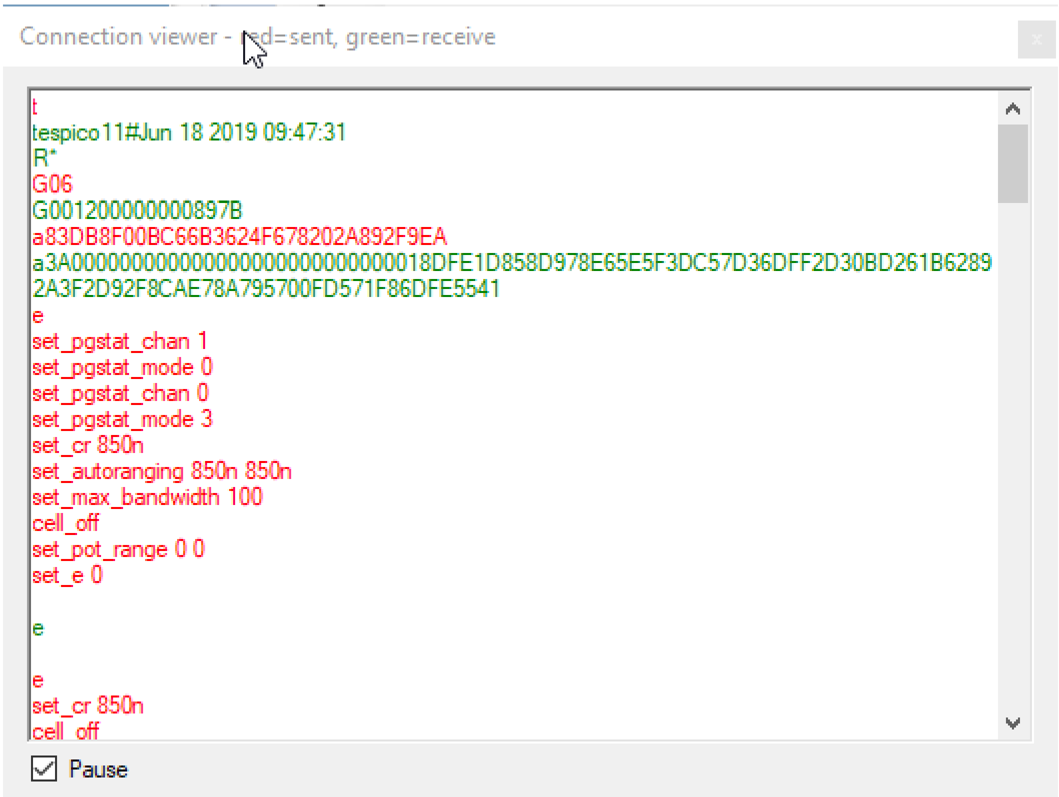
2.2 Connection viewer

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

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


Connection viewer - red=sent, green=receive

```
t
tespico11#Jun 18 2019 09:47:31
R*
G06
G001200000000897B
a83DB8F00BC66B3624F678202A892F9EA
a3A000000000000000000000000000018DFE1D858D978E65E5F3DC57D36DFF2D30BD261B6289
2A3F2D92F8CAE78A795700FD571F86DFE5541
e
set_pgstat_chan 1
set_pgstat_mode 0
set_pgstat_chan 0
set_pgstat_mode 3
set_cr 850n
set_autoranging 850n 850n
set_max_bandwidth 100
cell_off
set_pot_range 0 0
set_e 0

e

e
set_cr 850n
cell_off
```

☑ Pause

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 newline. See section "Error codes" for more information about errors.

EmStat Pico Firmware version 1.3.XX uses MethodSCRIPT version 1.3.

2.4 Communication modes

The device can be in two communication modes:

1. Idle mode
2. Script execution mode

2.5 Idle commands overview

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 memory	3.7
"Lmscr"	Retrieve MethodSCRIPT from non-volatile memory to RAM	3.8
's'	Set device into sleep-mode (deprecated)	3.9
'i'	Get device serial number	3.10
'v'	Get MethodSCRIPT version	3.11
"dlfw"	Enter bootloader mode. Invalidates current firmware.	3.12
"fs_dir"	Print the content of the directory	3.13.1
"fs_get"	Print the content of a file	3.13.2
"fs_put"	Store a file	3.13.3
"fs_del"	Remove a file or directory (recursively)	3.13.4
"fs_info"	Display storage medium information (free/used/total space)	3.13.5
"fs_format"	Format the file storage medium	3.13.6
"fs_mount"	Mount the filesystem	3.13.7
"fs_unmount"	Unmount the filesystem	3.13.8
"fs_clear"	Removes all files and folders from the storage medium	3.13.9
"CC"	Get runtime capabilities	3.14.1
"CM"	Get MethodSCRIPT capabilities	3.14.2

Table 1; Idle commands

2.6 Script execution commands overview

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’ – Version

Print the firmware version data of the device.

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

Command format
t\n

Response format
The first character is the echo of the command character ‘t’ followed by
espicoxxxx#mmm dd yyyy hh:mm:ss\n
R*\n

Where:

espico	Emstat Pico identifier
xxxx	represents the firmware version without the decimal points. When xxxx = 1304, the firmware version is 1.3.04.
#	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

Note that the day is preceded by a space character if it has a single digit and time is preceded by a ‘0’.

Example
t\n

Example response
tespico12#Jun 7 2020 09:37:02\n
R*\n

3.2 ‘S’ – Set register

Sets the value of a register. See section “Registers” for detailed information.

Command format

Sxxy..y\n

Where:

xx 2 digit hexadecimal register identifier

y..y Hexadecimal digits representing the value of the register to be set. The number of digits depends on the register.

Response format

S\n

Example

S0801\n Set register 08 (autorun) to value 01 (enabled)

Example response

S\n

3.3 ‘G’ – Get register

Gets the value of a register. See section “Registers” for detailed information.

Command format

Gxx\n

Where:

xx 2 digit hexadecimal register number

Response format

Gy..y\n

Where:

y..y Hexadecimal digits representing the value of the register. The number of digits depends on the register.

Example

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

Example response

G01\n Autorun is enabled

3.4 ‘l’ – Load MethodSCRIPT

Load and parse a MethodSCRIPT to RAM. The end of the script is indicated by an empty line containing only a newline ‘\n’ character. If no error was returned during loading, the script can be executed by the ‘r’ command (see section 3.5).

Command format
```
l\n
<script>
\n
```

Where:
<script>   MethodSCRIPT to load

Response format
```
l\n
```

3.5 ‘r’ – Run MethodSCRIPT

Execute a loaded MethodSCRIPT.

Command format
```
r\n
```

response format
```
r\n
<script output>
\n
```

3.6 ‘e’ – Load and run MethodSCRIPT

Parse and load a MethodSCRIPT to RAM, then execute it if no errors have occurred during parsing.

Command format
```
e\n
<script>
\n
```

Where:
<script>   MethodSCRIPT to load, terminated by a ‘\n’ character on an empty line

Response format
```
e\n
<script output>
\n
```

Example
```
e\n
send_string “hello world”\n
\n
```

Example output
```
e\n
Thello world\n
\n
```

3.7 ‘Fmscr’ – Store MethodSCRIPT in NVM

Store a loaded MethodSCRIPT to non-volatile memory.

Command format
Fmscr\n

Response format
F\n

3.8 ‘Lmscr’ – Load MethodSCRIPT from NVM

Loads a stored MethodSCRIPT to RAM from non-volatile memory. It can now be started with the ‘r’ command.

Command format
Lmscr\n

Response format
L\n

3.9 ‘s’ – Hibernate (deprecated)

Sets the device into sleep (hibernate) mode. The device will wake-up when the host sends data (commands) to the device or when the “Wake / GPIO_7” pin is brought low.

Note: This command is deprecated and may be removed in feature releases, use the MethodSCRIPT “hibernate” command instead.

Command format
s\n

Response format
s\n

3.10 ‘i’ – Get device serial

Gets the device serial number.

Command format
i\n

Response format
iSSSSSSSS\n

Where:

SSSSSSSS	8-character serial number

3.11 'v' – Get MethodSCRIPT version

Gets the MethodSCRIPT version.

Command format
v\n

Response format
vVVVV\n

Where:

VVVV	4-digit hexadecimal MethodSCRIPT version

3.12 'dlfw' – Enter bootloader

Resets the device in bootloader mode. A side-effect of this command is that the current firmware will be erased, meaning new firmware must always be uploaded after calling this command.

Command format
dlfw\n

Response format
d\n

3.13 File browser commands

The EmStat Pico can read and write data from/to a supported storage medium. The EmStat Pico supports most HC SD cards or the TC58CVG2S0 flash storage IC through an SPI interface. The file browser interface is provided to interact with this storage medium and supports data in ASCII encoding. Note that MethodSCRIPT also has the capability to interface with the filesystem, allowing the streaming of measurement data to the file system.

3.13.1 'fs_dir' – Show directory

The command "fs_dir <PATH>\n" prints all names of files and directories in the directory indicated by the parameter PATH. Files in subdirectories of the given path will also be printed. The EmStat Pico will respond with an "f\n" followed by the lines containing the files and directories. The list is terminated by an empty line. The format for each line is:
"DATE TIME;TYPE;NAME". Note that the values of "DATE" and "TIME" are separated using a space and the other field use a semicolon for this purpose.

Example
```
fs_dir /measurements\n
```

Prints the names the files and folders in the "/measurements" directory.

Example output
```
f\n
2019-12-31 11:34:13;DIR;0;measurements\n
2019-12-31 11:34:18;FIL;0;log.txt\n
2019-12-31 11:34:23;FIL;0;info.txt\n
2019-12-31 11:34:27;FIL;0;error_codes.csv\n
\n
```

3.13.2 'fs_get' – Get file content

The command 'fs_get <PATH>\n' prints "f\n", followed by the contents of the requested file. The end of the file is indicated with a file separator (ASCII) character (0x1C).

Example
```
fs_get /measurements/my_lsv_file.data\n
```

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

Example output
```
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
\x1C
```

Note: the file browser does not support the transmission of binary files.
Note2: the EmStat Pico transmits the data as fast as it can and will not wait for the host-system.

3.13.3 'fs_put' – Store file content

The command 'fs_put <PATH>\n<CONTENT>\x1C' stores the content of a file to the specified path. The end of the file is indicated with a file separator (ASCII) character (0x1C).

If a file with the same name already exists, an error is generated.

Example
```
fs_put /hello_world.txt\nHello World!\x1C
```

Stores a file containing the string "Hello World!" to the path "/hello_world.txt".

Example output
```
f\n
```

Note: the filebrowser does not support the transmission of binary files.

3.13.4 'fs_del' – Remove file/directory

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

Example
```
fs_del /log.txt\n
```

Removes the file "/log.txt".

Example output
```
f\n
```

3.13.5 'fs_info' – Get SD-card information

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

Example
```
fs_info\n
```

Example output
```
f\n
used:192kB free:7878464kb total:7878656kb\n
```

3.13.6 'fs_format' – Format SD-card

This command formats the SD-card with the FAT-filesystem. As a side-effect all content of the storage medium is erased.

Example
```
fs_format\n
```

Example output
```
f\n
```

Note: This is not the preferred way to clear an SD-card. For that use the 'fs_clear' command.
Note2: The formatting procedure can take some time. It will print "Format successful" when done
Warning: This operation cannot be undone.

3.13.7 'fs_mount' – Mount filesystem

Mounts the filesystem, as read from the storage medium. Ensure the file storage medium GPIO pins are configured as such. See the MethodSCRIPT command "set_gpio_cfg" for more information.

Example
fs_mount\n

Example output
f\n

3.13.8 'fs_unmount' – Unmount filesystem

Unmounts the filesystem. This can be used to re-mount the filesystem, in combination with "fs_mount".

Example
fs_unmount\n

Example output
f\n

3.13.9 'fs_clear' – Remove all files and directories

This command removes all files and directories on the SD-card.

Example
fs_clear\n

Example output
f\n

Warning: This operation cannot be undone.

3.14 'CC' and 'CM'

The CC (communication capabilities) and CM (MethodSCRIPT capabilities) commands return a list of supported commands for the EmStat. These capabilities are represented as bit fields in Hexadecimal format (256 bits, one per command). Each bit is tied to a specific command, if the feature is enabled then the corresponding bit is high. These bit fields are consistent across devices and take any licensing into account.

3.14.1 ‘CC’ – Runtime capabilities bit fields

Bit number	Command string	Command name
0	-	RESERVED
1	“t”	Version
2 - 31	-	RESERVED
32	“CC”	Communication capabilities
33	“CM”	MethodSCRIPT capabilities
34	“S”	Set register
35	“G”	Get register
36	“l”	Load script
37	“r”	Run script
38	“e”	Execute script
39	“dlfw”	Update firmware
40	-	RESERVED
41	“Wnvm”	Write NVM
42	“Rnvm”	Read NVM
43	“Fmscr”	Write MethodSCRIPT to non-volatile memory
44	“Lmscr”	Load MethodSCRIPT from non-volatile memory
45	“y”	Set date and time
46	“s”	Sleep (deprecated)
47	-	RESERVED
48	“i”	Get serial
49	“v”	MethodSCRIPT version
50	“x”	Self-test
51	“fs_dir”	File browser get directory
52	“fs_get”	File browser get file
53	“fs_put”	File browser write file
54	“fs_del”	File browser delete file
55	“fs_info”	File browser get filesystem info
56	“fs_format”	File browser format filesystem
57	“fs_mount“	File browser mount filesystem
58	“fs_unmount”	File browser unmount filesystem
59	“fs_clear”	File browser delete all files and directories
60	“m”	Get multi-device serial
96	“h”	Hold/pause MethodSCRIPT
97	“H”	Resume MethodSCRIPT
98	“Z”	Abort MethodSCRIPT
99	“Y”	Skip MethodSCRIPT instruction

3.14.2 ‘CM’ – MethodSCRIPT capabilities bit fields

Bit number	Command string	Description
0	-	RESERVED
1	“var”	
2	“array”	
3	“store_var”	
4	“copy_var”	
5	“add_var”	
6	“sub_var”	
7	“mul_var”	
8	“div_var”	
9	“set_e”	
10	“set_int”	
11	“await_int”	
12	“wait”	
13	“loop”	
14	“endloop”	
15	“breakloop”	
16	“if”	
17	“else”	
18	“elseif”	
19	“endif”	
20	“get_time”	
21	“meas”	
22	-	RESERVED
23	“meas_loop_lsv”	
24	“meas_loop_cv”	
25	“meas_loop_dpv”	
26	“meas_loop_swv”	
27	“meas_loop_npv”	
28	“meas_loop_ca”	
29	“meas_loop_pad”	
30	“meas_loop_ocp”	
31	“meas_loop_eis”	
32	“set_autoranging”	
33	“pck_start”	
34	“pck_add”	
35	“pck_end”	
36	“set_max_bandwidth”	
37	“set_cr”	
38	“cell_on”	

39	“cell_off”	
40	“set_pgstat_mode”	
41	“send_string”	
42	“set_pgstat_chan”	
43	“set_gpio_cfg”	
44	“set_gpio_pullup”	
45	“set_gpio”	
46	“get_gpio”	
47	“set_pot_range”	
48	“on_finished:”	
49	“set_poly_we_mode”	
50	“file_open”	
51	“file_close”	
52	“set_script_output”	
53	“array_get”	
54	“array_set”	
55	“i2c_config”	
56	“i2c_write_byte”	
57	“i2c_read_byte”	
58	“i2c_read”	
59	“i2c_write”	
60	“i2c_write_read”	
61	“hibernate”	
62	“abort”	
63	“timer_start”	
64	“timer_get”	
65	“set_range”	
66	“set_range_minmax”	
67	“meas_loop_cp”	
68	“set_i”	
69	“meas_loop_lsp”	
70	“meas_loop_geis”	
71	“int_to_float”	
72	“float_to_int”	
73	“bit_and_var”	
74	“bit_or_var”	
75	“bit_xor_var”	
76	“bit_lsl_var”	
77	“bit_lsr_var”	
78	“bit_inv_var”	
79	“set_channel_sync”	
80	“set_acquisition_frac”	

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' – Halt

Sending "h\n" to the device holds a running MethodSCRIPT

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' – Resume

Sending "H\n" to the device resumes a halted MethodSCRIPT

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’ – Abort

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.

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’ – Skip

Sending “Y\n” to the device breaks the execution of the current MethodSCRIPT loop after the current iteration of the loop has finished.

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 retrieve information, configure the device, or perform rarely used actions. Some registers are write protected at startup and must be unlocked before use, see section "5.2 '02' – Register permissions". The data length of each register is given in bytes of represented data. This data is communicated in hexadecimal notation, using 2 characters per byte.

Value	Permission level mode		Length (Bytes)	Description	See section
	Basic	Advanced			
0x01	Read only	Read / Write	4	Peripheral configuration	0
0x02	Read / Write	Read / Write	4	Register permissions	5.2
0x04	Read only	Read only	8	License register	5.3
0x05	Read only	Read only	16	Unique ID	5.4
0x06	Read only	Read only	8	Device serial	5.5
0x08	Read only	Read / Write	1	Autorun enable/disable	5.6
0x09	Read only	Read / Write	4	Advanced options	5.7
0x0A	Read / Write	Read / Write	4	Communication data rate limit	5.8
0x0B	Write only	Write only	4	Reset device	5.9
0x83	None	Write only	4	Auto calibration	5.10
0xA0	Read only	Read / Write	4	Low speed TIA 10M CH0 gain	5.11
0xA1	Read only	Read / Write	4	Low speed TIA 10M CH0 offset	5.11
0xA2	Read only	Read / Write	4	Low speed TIA 10M CH1 gain	5.11
0xA3	Read only	Read / Write	4	Low speed TIA 10M CH1 offset	5.11
0xA4	Read only	Read / Write	4	High speed TIA 10M gain	5.11
0xA5	Read only	Read / Write	4	High speed TIA 10M offset	5.11
0xA6	Read only	Read / Write	4	High speed TIA 1M gain	5.11
0xA7	Read only	Read / Write	4	High speed TIA 1M offset	5.11

5.1 ‘01’ – Peripheral configuration

Reads / writes the peripheral configuration as a bitmask from / to non-volatile memory. Support for external peripherals can be enabled here. Pins for peripherals that are not enabled can be used as GPIO pins. All peripherals default to GPIO. Multiple peripherals can be enabled at the same time by adding the hexadecimal values.

Value	Name	Description
0x00000020	Output 1.8V reference enable	When enabled, output 1.8V reference to the ANALOG_IN_2 pin. ANALOG_IN_2 can no longer be used as an input.
0x00000040	Output cell on/off status enable	When enabled, output cell on/off status on GPIO6. Cell on outputs a logic 0, cell off output a logic 1. GPIO6 can no longer be used as GPIO.
0x00000080	External RTC (S-35390A) init enable	When enabled the RTC (S-35390A) will be initialized after power on. This stops the RTC generating a 1Hz signal from potentially interfering with the EmStat Pico wake-up signal.
0x00000100 .. 0x80000000	Reserved	Reserved for future use. Do not change.

Example
“S0100000020\n” sets the peripheral configuration register. This will enable the 1.8V reference.

Example response
S\n

5.2 ‘02’ – Register permissions

By default, most registers are write protected to prevent accidental writes. This register can be used to disable the write protection. It is advised to turn the write protection back on when access to write protected registers is no longer required.

Level	Key	Description
Basic	0x12345678	Default configuration at startup. No access to registers non-volatile registers.
Advanced	0x52243DF8	Full access to all user changeable settings.

Example
“S0252243DF8\n” sets the permission level to advanced, allowing full access.

Example response
S\n

5.3 ‘04’ – License register

Request the licenses programmed into this EmStat Pico. For more information contact PalmSens.

Example
“G04\n” gets the license register.

Example response
Gxxxxxxxxxxxxxxxx\n

Where:

xxxxxxxxxxxxxxxx	= 16 hexadecimal digit license code

5.4 ‘05’ – Unique ID

Reads the unique ID for this device.

Example
“G04\n” gets the unique ID register.

Example response
Gxxxxxxxxxxxxxxxx\n

Where:

xxxxxxxxxxxxxxxx	= 32 hexadecimal digit unique ID code

5.5 ‘06’ – Device serial

Contains the device serial number.

Example
“G06\n” gets the serial number of the device.

Example response
Gttyybbbbnnnnnnnn\n

Where:

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.

5.6 ‘08’ – Autorun

Contains the autorun setting. If set to 1, the MethodSCRIPT stored in non-volatile memory will be loaded and executed on startup. When the script ends, the EmStat Pico returns to its normal behavior.

Example
“S0801\n” sets the autorun register to 01 (autorun enabled)

Example response
S\n

5.7 ‘09’ – 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.

The Pico currently has a device specific option bit for “Extended voltage range”. Enabling this reduces the accuracy of measured currents and is not recommended. To enable it write “00000001” to this register. Write “00000000” to disable it.

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 EmStat Pico protocol – CRC 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.

Example
“S0900000001\n” will enable the “extended voltage range” option (and disable the CRC16 mode).

Example response
S\n

5.8 ‘0A’ – Communication data rate limit

This register allows limiting the maximum bytes per second that is sent by the device. This is independent from the UART baudrate. This can be useful when no flow control mechanism is used with UART and the host cannot keep up with the data rate defined by the baudrate.

Example
“S0A00000400\n” sets the maximum data rate to 1024 bytes per second.

Example response
S\n

5.9 ‘0B’ – Reset device

Writing 0x93628ADE to this register will initiate a software reset of the device. Note that this command will not return a newline if the reset is successful.

Example
“S0B93628ADE\n” resets the device.

Example response
S

5.10 ‘83’ – Auto calibration

Writing 0x4321ABCD to this registers will initiate the built-in auto calibration sequence. This sequence requires the WE to be unconnected. The calibration will take up to 60 seconds. This calibration will not affect the 10M and 1M calibrations accessible by the registers below.

Warning: PalmSens does not recommend re-calibrating factory calibrated devices.

Example
“S834321ABCD\n” initiates auto calibration.

Example response
S\n

5.11 Manual calibration registers

The EmStat Pico has a few calibrations that cannot be done automatically, these calibrations are accessible through their respective registers.

Note: High precision resistors and measurement equipment are needed to perform these calibrations.

The following functions can be used to convert the register value to gain and offset values:

Calibration type	Function
Low speed TIA gain	Factor = 1 / (reg / 0x4000)
Low speed TIA offset	Offset (V) = -reg / 4 * 5.5999756e-5
High speed TIA gain	Factor = 1 / (reg / 0x4000)
High speed TIA offset	Offset (V) = (-reg - 0x4000) / 4 * 5.5999756e-5

The following calibration registers are available:

0xA0	Low speed TIA 10M CH0 gain
0xA1	Low speed TIA 10M CH0 offset
0xA2	Low speed TIA 10M CH1 gain
0xA3	Low speed TIA 10M CH1 offset
0xA4	High speed TIA 10M gain
0xA5	High speed TIA 10M offset
0xA6	High speed TIA 1M gain
0xA7	High speed TIA 1M offset

6 CRC16 protocol extension

6.1 Introduction

For certain applications of the EmStat Pico, 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 document describes an extension that adds CRC and sequence-ID fields to the EmStat Pico communication protocol to allow for verification of the received data. The CRC-extension will be selectable in the EmStat Pico's non-volatile 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 EmStat 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 EmStat, the EmStat 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 EmStat does not expect this.

The EmStat Pico 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 used:

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

6.3.1 Example command without CRC-extension

t\n	From host
tespico11#Jun 18 2019 09:47:31\n	From Pico
R*\n	From Pico

6.3.2 Example command with CRC-extension

t0A9524\n	From host
<0A>454FBA\n	From Pico
tespico12#Apr 23 2020 15:41:4646DA41\n	From Pico
D*47EE4F\n	From Pico

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

6.3.3 MethodSCRIPT example command without CRC-extension

e\n	From host
e	From Pico (note: no \n)
send_string “Hello World!”\n	From host
\n	From host
THello World!\n	From Pico
\n	From Pico (\n to close command)

6.3.4 MethodSCRIPT example command with CRC-extension

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

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 v1_3.pdf".
See https://www.palmsens.com/knowledgebase-article/methodscript

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 (EmStat Pico)
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	An 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
0040	STATUS_FLASH_NOT_SUPPORTED	Memory module not supported.
0041	STATUS_FS_INVALID_FORMAT	Filesystem memory format not recognized or supported
0042	STATUS_REGISTER_ACCESS_DENIED	This register is locked for current permission

		level
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
FFFF	STATUS_ASSERT_FAIL	An unexpected error has occurred. A reset is required.

Table 3; Error codes

8 Version changes

Version 1.2

- Added filebrowser commands
- Updated error codes table
- Added extra registers

Version 1.3

- Added CRC16 extension (option bits, error values and chapter)
- Removed “cali” command (replaced by register)
- Added “fs_put” command
- Updated fs_* commands to include extra acknowledgement newline
- Fixed broken MethodSCRIPT link
- Updated registers
- Updated error codes
- Updated capability definitions
- Updated version string with extra “patch” field