PalmSens
Compact Electrochemical Interfaces

EmStat Pico communication protocol V1.1

Rev. 01

Contents

PalmSens

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 can be started by sending a MethodSCRIPT, more information about MethodSCRIPT can be found here:

https://embed.palmsens.com/knowledgebase_category/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
MethodSCRIPT:	PalmSens human readable script format

2 Communication settings

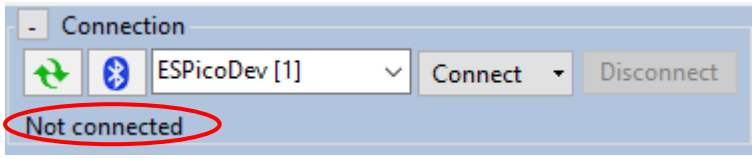
The EmStat Pico communicates using 3.3V UART (Serial Port) with the following settings:

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

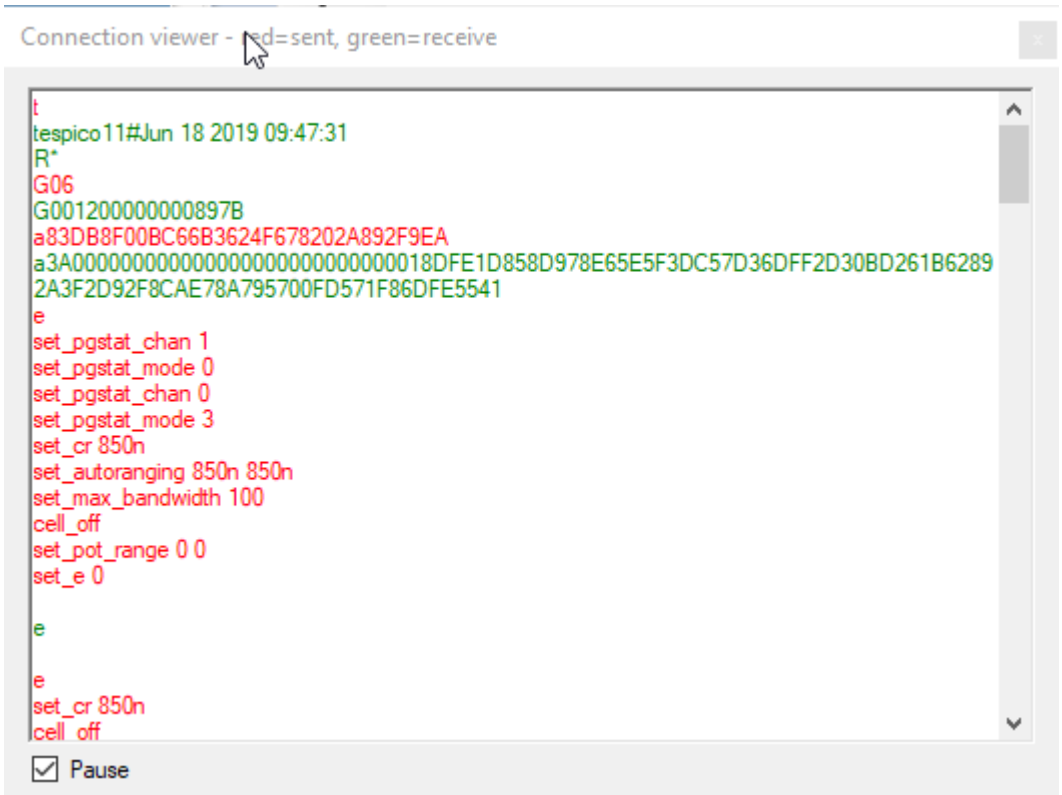
2.1 Connection viewer

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



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.

2.2 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 "Response:
S\n

Error codes" for more information about error.

2.3 Communication modes

The device can be in two communication modes:

1. Idle mode
2. Script execution mode

The EmStat Pico is controlled using the following characters or commands (without quotes):

2.4 Commands overview

Idle commands

Command	Description	See section
't'	Get the device firmware version	2.5
'S'	Set register value	2.6
'G'	Get register value	2.7
'l'	Load (parse) MethodSCRIPT to RAM	2.8
'r'	Run (execute) loaded MethodSCRIPT	2.9
'e'	Load and execute MethodSCRIPT (same as 'l' followed by 'r')	2.10
"cali"	Calibrate the device	2.11
"cali clear"	Clear calibration data of the device	2.12
"Fmscr"	Store loaded MethodSCRIPT to non-volatile (Flash) memory	2.13
"Lmscr"	Retrieve MethodSCRIPT from non-volatile memory to RAM	2.14
's'	Set device into sleep-mode	2.15
'i'	Get device serial number	2.16
'v'	Get MethodSCRIPT version	2.17

Table 1; Idle commands

Script execution commands

Command	Description	See section
'h'	Hold execution of the running MethodSCRIPT	2.18
'H'	Resume execution of the halted MethodSCRIPT	0
'Z'	Abort execution of the running MethodSCRIPT	2.20
'Y'	Skip execution of current (or next?) MethodSCRIPT command	2.21

Table 2; Script execution commands

2.5 '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"

Example:
Send "t\n" response with:

"tespico11#Jun 18 2019 09:47:31\n"
"R*\n"

Decoding
The first character is the echo of the command character 't' followed by
"espicoxx#mmm dd yyyy hh:mm:ss\n"
"R*\n"

espico	Emstat Pico identifier
xx	represents the firmware version without the decimal point, e.g. when xx = 11, the firmware version is 1.1.
#	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

2.6 'S' command

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

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.

2.7 'G' command

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

Example:

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

Response:

Gyy.yy	;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.

2.8 '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, then the script can be executed by the 'r' command (see section 2.9).

Example (newline ("\n") characters are omitted):
l
send_string "hello world"

2.9 'r' command

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

2.10 '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.

Example (newline ("\n") characters are omitted):
e
send_string "hello world"

2.11 'cali' command

Sending "cali\n" to the device performs a self-calibration of the device. This can take up to a minute. Note: All electrodes need to be disconnected from the device prior to the command.

2.12 'cali clear' command

Sending "cali clear\n" to the device clears all calibration data. It is not recommended to perform measurements on an uncalibrated device.

2.13 'Fmscr' command

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

2.14 '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.

2.15 ‘s’ command

Sending “s\n” to the device brings the device into sleep (hibernate) mode. The device will wake-up when the host sends data (commands) to the device. It can also be woken up through the “Wake / GPIO_7” pin if it is configured as “Wake” pin.

2.16 ‘i’ command

Sending “i\n” to the device gets the device serial number.

Example:
Send “i\n” response with:

“iEP1CA8BR”

Decoding
The first character is the echo of the command character ‘i’ followed by the 8-character serial number.

2.17 ‘v’ command

Sending “v\n” to the device gets the MethodSCRIPT version.

Example:
Send “v\n” response with:

“v0002”

Decoding
The first character is the echo of the command character ‘v’ followed by the 4-digit hexadecimal version number.

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.

2.18 ‘h’ command

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 current command
   pck_add c
   pck_end
endloop
on_finished:
cell_off
```

2.19 'H' command

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
```

2.20 '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.

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
```

2.21 'Y' command

Sending "Y\n" to the device skips the execution of the current MethodSCRIPT loop after the next 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
```

3 Registers

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

Value	access mode	Description	See section
0x01	Read / Write	Peripheral configuration	3.1
0x04	Read only	License register	0
0x07	Read only	Hardware revision	3.3
0x08	Read / Write	Autorun enable/disable	3.4
0x09	Read / Write	Advanced options	3.5

3.1 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
0x00000001	Shutdown pin enable	Not Implemented yet
0x00000002	Enable SD Card enable	Not Implemented yet
0x00000004	SD card chip detect pin (CD) enable	Not Implemented yet
0x00000008	Wake up pin enable	Not Implemented yet
0x00000010	I2C enable	Not Implemented yet
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..0x80000000	Reserved	Reserved for future use. Set to 0.

Example
"S0100000023\n" sets the peripheral configuration register. This will enable the shutdown pin, SD Card and 1.8V reference.

Response:
S\n

Where:
S = echo of 'S' command
\n = End of command

3.2 License register

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

Example
"G04\n" gets the license register.

Response:
Gxxxxxxxxxxxxxxxx\n

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

3.3 Hardware revision

Contains the hardware revision version.

Example
"G07\n" gets the hardware revision of the device.

Response:
Gxx\n

Where:
G = echo of 'G' command
xx = 2 hexadecimal digit serial number
\n = End of command

3.4 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)

Response:
S\n

Where:
S = echo of 'S' command
\n = End of command

3.5 Advanced options

Contains the advanced option setting bitmask. The only option currently available is the “extended voltage range” option. 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.

Example
“S0900000001\n” will enable the “extended voltage range” option.

Response:
S\n

4 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 v?_?.pdf”.
See https://embed.palmsens.com/knowledgebase/methodscript-documentation/

The errors applicable to online communication (non-MethodSCRIPT) are highlighted in yellow.

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
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
7FFF	STATUS_FATAL_ERROR	A fatal error has occurred, the device must be reset

Table 3; Error codes