PIKK-Systems

Elektronikentwicklung

PIKKSENSE

Device User Manual

Version 0.3

Document History

Version	Date	Changes	State
v0.1	25.10.2023		Preview for internal use
v0.2	26.10.2023	Button differences PS-250 vs. PS-280	Preview for internal use
v0.3	19.12.2023	Exchange mode and OTA updates	Preview for internal use

Table of Contents

1 Legal

PIKK-Systems GmbH is not liable for technical or editorial errors or omissions in this document. Furthermore, it assumes no liability for damages directly or indirectly resulting from the provision, performance, or use of this material.

We reserve the right to make content changes to this document without prior notice. The information provided in this publication is provided without warranty of accuracy and completeness. In particular, this information does not imply any guaranteed properties. The user assumes all risks arising from the use of this information.

We would like to point out that the software and hardware designations and brand names used in the manual are generally protected by trademark, brand, or patent rights of the respective companies.

2 Symbols

! GEFAHR	This symbol indicates a safety notice. You are being alerted to an immediate danger to life or health. ➔ The arrow signifies a precautionary measure to avert this danger.
ACHTUNG	This symbol indicates a safety notice. You are being alerted to a potentially impending danger to life or health. ➔ The arrow signifies a precautionary measure to avert this danger.
! WARNUNG	This symbol indicates a safety notice. You are being alerted to a potentially hazardous situation to life or health. ➔ The arrow signifies a precautionary measure to avert this danger.
! VORSICHT	This symbol indicates a safety notice. You are being alerted to a danger to the product. ➔ The arrow signifies a precautionary measure to avert this danger.
Hinweis	Important Information: This symbol indicates information that may be helpful or necessary for handling the product. This includes references to further information.

3 Overview

The devices in the PS-200 family are ultra-low-power devices optimized for long-lasting battery operation. There is a wide range of sensor and communication options available.

Custom options can also be implemented relatively quickly due to the modular housing design.

4 Options

Following options are available so far:

Device Name	Sensors
PS-221	Temperature, rel. humidity
PS-240	Pulse counter, contact monitoring / dry contact, temperature, rel. humidity
PS-250	Temperature (2x PT1000), and connectivity for max. 8x PS-251 extenders
PS-251	Extender for PS-250 - 8x Temperature sensor (1-wire) each
PS-280	Temperature, rel. humidity, forehead temperature via IR thermopile array scan, people counting, VOC, CO2

5 Safety Regulations

5.1 Safety Regulations and Compliance Declarations

The product complies with the essential requirements of the applicable European directives. The conformity declaration can be provided upon request.

5.2 Hazardous Substances

The product does not contain any substances listed in the Regulation on the Protection against Hazardous Substances, published in the Federal Law Gazette I S.1782 (Gefahrstoffverordnung, abbreviated as GefStoffV).

5.3 Product Disposal

The product and, if applicable, the components marked with the waste bin symbol fall within the scope of the Electrical and Electronic Equipment Act (ElektroG2).

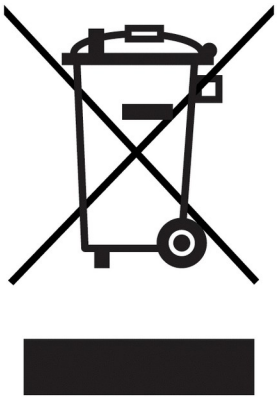
The ElektroG2 implements the following EU directive: 2012/19/EU (WEEE) for Waste Electrical and Electronic Equipment.

The labeling of the respective products is done using the symbol specified in EN 50419.

At the end of the product's life cycle, it must not be disposed of with regular household waste. Disposal through municipal collection points for electronic waste is also not permitted.

For environmentally friendly disposal or recycling, PIKK-Systems GmbH has developed a disposal concept and assumes the obligations of take-back and disposal in accordance with the ElektroG on behalf of the manufacturer.

Please contact PIKK-Systems GmbH for the disposal of the product.



6 Delivery Scope, Accessories, Spare Parts

The standard delivery scope includes

- 1x PS-2xx

Following accessories are available

- External USB power adapter
- Lithium batteries

[TBD]

7 Structure, Controls and Display Elements

[TBD]

8 Configuration

8.1 Configuration via USB interface (console mode)

8.1.1 Setup Terminal and Connection (Tera Term)

For easiest interaction with the device, it is recommended to use a terminal emulator such as Tera Term, which also supports the execution of scripts for automatic device configuration.

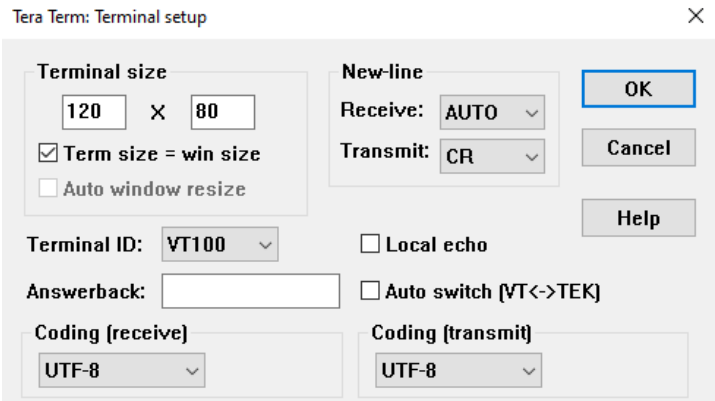
The executable file can be downloaded directly on the homepage *https://ttssh2.osdn.jp/*.

Terminal Configuration:

Once installed and opened up, Tera Term proposes to start a new connectversion_xzy/ion, but this we can dismiss by pressing the ***"Cancel"*** button for now.

For best visual results in the terminal, select ***"Setup/Terminal…"*** from the menu for configuration.

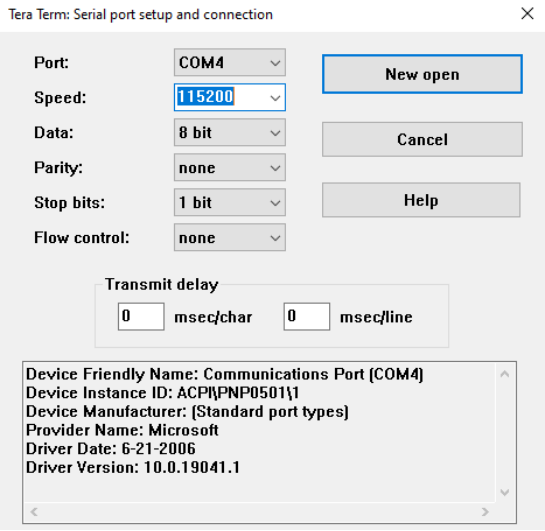
Here the **terminal size** should be increased and also the **new-line receive** be set to ***"AUTO"***:



Serial Connection Configuration:

With the menu option *"Setup/Serial port…"* the setting of the serial port can be specified, be sure to use **115200** instead of the default 9600 as the speed stetting/baud rate, and to select the correct device port, if more than one COM port is active.

Afterwards press ***"New open"*** to start the session and connect to the device (or ***"New setting"*** to save the setting, if a connection is already running):



Save/Restore Setup:

Once everything is configured, this setup can be stored, so that it is available again when Tera Term is started the next time, otherwise the default setup is loaded again.

For this select the option ***"Setup/Save setup…"*** from the menu and choose a place for the ***"*.ini"*** configuration file.

If you choose to replace the ***"TERATERM.ini"*** file while saving, then this will be the new default configuration. When using a different file, then at next launch of Tera Term the configuration has to be loaded manually by choosing the option ***"Setup/Restore setup…"*** from the menu.

Starting/Closing a Connection:

If the default configuration has been changed, then the device should automatically connect once Tera Term is started. Otherwise following options are available:

- For creating a new connection, select ***"File/New connection…"*** from the menu (or use keyboard shortcut ***"Alt+N"***
- For terminating a connection, do either option:
  - disconnect the session by selecting ***"File/Disconnect"*** or using the keyboard shortcut ***"Alt+I"***
  - or close Tera Term completely by selecting ***"File/Exit"*** or using the keyboard shortcut ***"Alt+Q"***

8.1.2 Disconnecting the Device

If a running device is disconnected from a power source, it is possible to corrupt single files or also the whole partition. For this a special shutdown procedure should be followed, to ensure that any log files are available for further inspection.

1. **Never disconnect the device before first terminating the serial connection (via Tera Term, PuTTY, etc.)**

2. **Before disconnecting the device from a power supply, ALWAYS perform a shutdown operation first:**
   a) Either:
      1. Press the ***"SERVICE"*** button **3x within 5 seconds** (if not connected to the computer)
      2. Use the command ***"halt"*** in the terminal (if a session is currently running in Tera Term/PuTTY)
   b) First wait for the **white LED** fading in and out a single time signaling the triggered shutdown process
   c) Then wait for the **blue LED** fading in and out a single time signaling the complete halting of the device (can take up to 60 seconds)
   d) The device is now safe to be disconnected from the power supply

8.1.3 Commands and Terminal Interaction

After connecting to the device per the serial connection, the output of the startup process and current operations can be seen, including the current firmware version printed on the top.

```
#######################################################
#### PSense OS V0.4.1.1395.e67227a.20230907_111809 ###
#######################################################
(I) [00:00:02] qstor.cpp::master_block_reset():198 | Cleared FLASH
(I) [00:00:02] qstor.cpp::master_block_reset():198 | Cleared RAM
(I) [00:00:02] esp32s3_int_flash.cpp::esp32s3_int_flash():22 | ESP32 internal storage
driver started
(I) [00:00:02] esp32s3_int_flash.cpp::esp32s3_int_flash():41 | Phy_info [name=user|
blk_sz=4096|blk_cnt=256|sz=1048576]

(...)

(I) [00:00:38] bg770.cpp::configuration_step_pre():323 | Auto bauding: OK
(I) [00:00:38] commdev_sock_at.cpp::echo():161 | ECHO off: OK
/ >
```

Most of the lines are information (I) in white text, warnings (W) in yellow and any possible errors (E) are in red. Other application output is printed in green.

Further it is possible to send commands to the device for configuration and log file access, the following line depicts the expected user input, while currently being in the root folder ("/"):

```
/ >
```

Similar to a Linux command line you can interact with it:

- type a command and press *"Enter"*
- use the "Up/Down Arrow" for recalling and editing previously used commands

Commands

Following commands are needed for basic operation:

settings	Print out all current settings to the console (further options and details on how to configure the settings are handled in the next chapter)
ls	List all files in current folder (default folder is the root "/" of the user partition)
cd [directory]	Change the working directory ("cd .." to switch to parent folder)
part [partition]	Change the current partition ("part user" for the user partition and "part log" for the log file partition)
cat	Print out a file to the console

Device Settings

The most important command is "settings" for inspecting and changing the device settings. Using following command without any further parameters will output the current settings to the console:

```
> settings
```

Example output:

```
settings [info|list|load|restore|set|store]
Module    Setting             RO   Value

CORE      BCMP                     1
CORE      CPORT               X    0
CORE      FLOG_MAXF                10
CORE      LOG_LEVEL                0
CORE      MSC                      4
CORE      MSI                      300
CORE      SERIAL              X    PSENSE-123456
CORE      SHELL_TO                 30
CORE      SHELL_TO_ENA             0
CORE      TRANSPORT                wifi
CORE      VERSION             X    0.4.1.1395.e67227a.20230907_111809
HUB       EXCH_CNT                 10
HUB       EXCH_MODE                1
HUB       EXCH_TIMER
HUB       EXCH_TO                  30
HUB       LIFETIME                 30
HUB       NTP_IP                   de.pool.ntp.org
HUB       NTP_PORT                 123
```

```
HUB       PROTOCOL                tcp
HUB       REMOTE_IP               test.mosquitto.org
HUB       REMOTE_PORT             1883
HUB       STORE_MAX               250
HUB       TSYNC                   1440
HUB       TSYNC_WAIT              1
HUB       TYPE                    mqtt
HUB       T_RETRY                 60
HUB       T_RETRY_MAX             43200
HUB       T_RETRY_MODE            1
MODEM     APN                     demoapn-02
MODEM     APN_PW
MODEM     APN_USER
MODEM     BANDS_LTE               [20]
MODEM     BANDS_NB                [8,20]
MODEM     CAT                     any
MODEM     OP                      0
MQTT      CLIENT_ID
MQTT      DIAG_CONN               0
MQTT      DIAG_SYS                0
MQTT      MAX_RETRY               4
MQTT      PL_SIG_ENA              0
MQTT      PL_SIG_PW               psense-client
MQTT      PW                      demo_pw
MQTT      QOS                     1
MQTT      TIMEOUT                 15
MQTT      TOPIC_DOWN              PS-280/down
MQTT      TOPIC_UP                PS-280/up
MQTT      USER                    demo_user
OTA       RESULT              X   0
OTA       UPDATE                  0
OTA       URL                     http://update.example.com/firmware/main.bin
RUNTIME   RSSI                X   -999
SEC       CA_PATH                 /sec/ca.pem
SEC       CC_PATH                 /sec/cc.pem
SEC       LOG_LEVEL               0
SEC       MODE                    0
SEC       TO_RD                   3
SEC       UK_PATH                 /sec/uk.key
SENSOR    HM_MODE                 pc
SENSOR    OT_DTO                  100
SENSOR    PC_ALERT                0
SENSOR    PC_THRES                15
SENSOR    PC_INC                  30
SIG       BOOT_AUR                1
SIG       BOOT_VIS                1
SIG       REG_VIS                 1
THRESH    PC_IN_ENA               0
THRESH    PC_IN_LO                0
THRESH    PC_IN_HI                1000
THRESH    PC_OUT_ENA              0
THRESH    PC_OUT_LO               0
THRESH    PC_OUT_HI               1000
THRESH    VBAT_VBAT_ENA           0
THRESH    VBAT_VBAT_LO            -1000
THRESH    VBAT_VBAT_HI            1000
WIFI      AP_BSSID
WIFI      AP_PW                   ****
WIFI      AP_SEC                  wpa2_psk
WIFI      AP_SSID                 demo_ap
WIFI      TX_POWER                15
```

This output shows all available settings (second column) and their respective values (fourth column). The third column "RO" shows if the settings/properties are read-only or can also be changed.

For logical separation, the settings are grouped into "modules" (first column), so when dealing with them this has to be specified each time:

CORE	Core system settings
HUB	IoT hub settings for the data exchange with your server
MODEM	LTE modem settings for internet access
MQTT	MQTT message and topic settings for the data up-/download
OTA	Configuration for the Over-the-Air update mechanism
RUNTIME	Runtime information that can be read out, e.g. RSSI of the antenna
SEC	Security and certificate related settings
SENSOR	Sensor settings
SIG	Signaling configuration for notification (LED, sound)
THRESH	Measurement threshold settings (person count, battery)
WIFI	Wi-Fi settings for internet access

Changes to the configuration can me made with following command structure:

```
> settings set [module] [setting] [value]
```

Following example with a correct and successful command that tells the system to use Wi-Fi as the transport method:

settings set core transport wifi	(I) [00:02:05] kernel.cpp::settings_store():4691 | Settings stored

If something went wrong with the command, then the system will let you know with an error. In this example a non-permitted value is tried to be set. Further the warning will also give you a hint on what values are permitted:

settings set core transport wrong_value	(E) [00:10:09] setting.cpp::assign():141 | value "wrong_value" out of range for setting "TRANSPORT" (W) [00:10:09] setting.cpp::assign():174 | Permitted: {"modem","wifi"}

If you want to directly check what valid values can be used, the settings information command can be used for this:

```
settings info [module] [setting]
```

Example:

```
settings info
core transport
```

```
TRANSPORT
---------
Info: Transport hardware selection
Allowed: {modem,wifi}
```

Following example illustrates how to set up the device for using your local Wi-Fi connection, first changing the transport type to use Wi-Fi and then setting the SSID and Password for the access point:

```
settings set core transport wifi
settings set wifi ap_ssid <SSID>
settings set wifi ap_pw <PASSWORD>
```

8.1.4 Automatic Configuration via USB and TTL Script

Optionally an automatic configuration of the device can be performed, for this a ***.ttl** file is needed in combination with Tera Term. A file may be provided by us for specific setups or can be created fresh and adapted to your needs.

The ***"*.ttl"*** file consists of a list of commands that will be executed after each other in the terminal. Possible commands for this are listed in the previous chapter.

Between each command a short sleep phase of 0.5 seconds should be used to ensure that the commands are all processed by the device before the next command is sent.

Example script that sets the Wi-Fi settings and operation mode parameters with 500 milliseconds pause in-between:

```
;example configuration script
sendln 'settings set wifi ap_ssid example_access_point'
mpause 500
sendln 'settings set wifi ap_pw example_password'
mpause 500
sendln 'settings set core msc 3'
mpause 500
sendln 'settings set core msi 60'
mpause 500
```

For executing the script:

1. Connection has to be up and running and the device be able to receive commands in the terminal
2. Select the option ***"Control/Macro"*** from the menu, then locate and open the ***"*.ttl"*** file from you Windows system
3. The script should now be performing all operations inside it and print out the progress inside the terminal window
4. Once the script has stopped, close the terminal session
5. Disconnect the device from computer

Remember to properly disconnect the device first before removing the USB connection!

8.2 Configuration via Bluetooth

[TBD]

8.3 Remote Configuration via Exchange Mode

The PIKKSENSE devices allow remote configuration via the configured hub connection (e.g. MQTT broker).

8.3.1 Exchange Mode Settings and Life Cycle

Following settings are relevant for the functioning of the exchange mode, here with an example configuration used below:

```
Module    Setting             RO   Value

CORE      MSC                      4
CORE      MSI                      60
...
HUB       TYPE                     mqtt
HUB       EXCH_CNT                 10
HUB       EXCH_MODE                1
HUB       EXCH_TIMER
HUB       EXCH_TO                  30
...
HUB       REMOTE_IP                mqtt.example.com
HUB       REMOTE_PORT              1883
...
MQTT      TOPIC_DOWN               ps-280/down
MQTT      TOPIC_UP                 ps-280/up
```

In this configuration with MSI=60 and MSC=4, the example device will send out one measurement data package about every 4 minutes (see chapter 9.1.1 Synchronous Mode) through the TOPIC_UP MQTT channel.

If EXCH_MODE=1, then the exchange mode will be activated in these both situations:

- directly once at start of the device after a cold boot
- every EXCH_CNT-times a data package has been sent to the hub (in this example 10x4min, so about every 40 minutes once)

EXCH_MODE=0 deactives this function completely.

EXCH_TO defines the exchange mode timeout duration in seconds (here e.g. 30s)

Once the PIKKSENSE device is active in the exchange mode, it will notify the server of its state with adding "exch: 1" in the JSON payload of the MQTT message:

```
{
  "serial": "PS280-123456",
  ...
  "exch": 1
}
```

Now the device will wait for EXCH_TO seconds for any message that is published to the TOPIC_DOWN channel (here "ps-280/down").

If it does not receive anything, it then leaves the exchange mode again without any further action and returns to the normal measurement operations.

But if a message is available, it will try to change the settings included in the payload and send back the result or any error code generated. In this case the EXCH_TO timeout counter will be reset again after every incoming message, so that the exchange mode duration is extended further and provides the possibility to perform multiple operations in sequence without any interruption.

8.3.2 Sending Configuration Data

Commands have to be sent in the following JSON format:

```
{
   "<MODULE>":[
      {
         "name":"<SETTING_NAME>",
         "value":<SETTING_VALUE>,
      },
      ...
   ],
   …
}
```

Same as per the console, the settings have to be specified on a module basis. If strings are passed, they should be escaped with quotes.

Here is a simple message with only one setting:

```
{
  "CORE": [
    {
      "name": "MSC",
      "value": 6
    }
  ]
}
```

And then also an example payload with three settings for two modules at the same time:

```
{
  "CORE": [
    {
      "name": "MSC",
      "value": 6
    },
    {
      "name": "MSI",
      "value": 120
    }
  ],
  "HUB": [
    {
      "name": "LIFETIME",
      "value": 200
    }
  ]
}
```

8.3.3 Return Values and Errors

Success

If all settings in the message were performed successfully, then the device will return a zero afterwards as the combined result, the examples above would return this:

```
{
  “ret”: 0
}
```

Error

If any of the operations failed, then the returned value will specify which operation had a problem.

Following example now has two wrong settings (MSC is not between 0 and 32 and MSI is a not a positive integer):

```
{
  "CORE": [
    {
      "name": "MSC",
      "value": 50
    },
    {
      "name": "MSI",
      "value": -1
    }
  ],
  "HUB": [
    {
      "name": "LIFETIME",
      "value": 200
    }
  ]
}
```

The return message would look like this then:

```
{
  "ret": 13,
  "s_err": "Module CORE: MSC -> value rejected;MSI -> wrong type"
}
```

Return value 13 is the corresponding response error code, and the "s_err" holds the two error messages resulting from our two wrong commands.

The two first settings were rejected, though the variable LIFETIME of the module HUB will have be stored successfully.

9 Operation

9.1 Operating Modes

9.1.1 Synchronous Mode

The basic operation logic of the PS devices is based on the ***measurement interval*** (**MSI**: value between 0 and 32, in seconds) and represents the shortest time between two deep sleep phases of the controller. These deep sleep phases are crucial for achieving long battery life.

Each time the device wakes up after an interval of MSI seconds, it performs its measurement collection.

```
settings info
core msi
```

```
MSI
---
Info: Measurement Interval - Time in [s] between two measurements
Min. value: 1
Max. value: 4294967295
```

Combined with this *measurement interval*, a second parameter ***measurement count*** (**MSC**) is used, to determine, how many measurements should be combined into a single message and then be sent out to the server.

So if MSC is greater than 1, the controller wakes up after MSI seconds, performs the relevant measurements, saves them, and then goes back to the deep sleep/power-saving mode unless with this last measurement it has collected MSC number of untransmitted measurements. In this case, all this stored data is combined into a message and sent to the server configured server.

```
settings info
core msc
```

```
MSC
---
Info: Measurement Count
Min. value: 0
Max. value: 32
```

Example:

With values **MSI=60** and **MSC=4**, the device will wake up every 60 seconds and gather one set of measurements and go back to sleep. On every 4th wakeup from sleep, a message will be sent to the server with one single transmission. Like this the messages are sent out every 4 minutes.

```
settings set core msi 60
settings set core msc 4
```

[TBD] [Message diagram / MSI, MSC]

9.1.2 Asynchronous Mode

Event-based triggers, such as detecting motion through the PIR sensor or counting people, are processed independently of the synchronous time settings and may be immediately sent if necessary and configured. For such notifications, there are also adjustable thresholds to customize the sending behavior.

Example:

```
settings info
sensor pc_thres
```

```
PC_INC
------
Info:
Min. value: 0
Max. value: 4294967295
```

[TBD]

9.2 Payload / Data Format

9.2.1 MQTT: JSON Format

When the message type is set to use the MQTT protocol, then the payload will be in text and use the JSON format.

```
settings set hub type mqtt
```

Example MQTT message for the case of only sending one set of measurements per message (with setting **MSC=1**):

```
{
  "serial": "PS280-123456", // Serial Number
  "unix_time": 1669620309,  // Unix-Timestamp
  "aht20_hum": 47.48,       // Humidity
  "aht20_tem": 19.7,        // Temperature
  "t_object": 30.83,        // FeverDetect – Object temperature
  "contrast": 15.12,        // FeverDetect - Contrast
  "std_dev": 0.56,          // FeverDetect – Standard deviation
  "reg": 1                  // Registration flag (will only be send on a cold start)
}
```

Example MQTT message for the case of sending multiple sets of measurements per message (with setting **MSC>1**, here e.g. MSC=2). Here the data points for a specific measurement are combined as an array, while here the newest data point is last one in the array:

```
{
  "serial": "PS280-123456",
  "unix_time": 1669620455,
  "aht20_hum": [ 47.22, 49.19 ],
  "aht20_tem": [ 19.58, 19.58 ],
  "t_object": [ 30.77, 30.8 ],
  "contrast": [ 15.02, 15.0 ],
  "std_dev": [ 0.65, 0.67 ]
}
```

9.2.2 CoAP / LwM2M

[TBD]

9.3 Log Files

9.3.1 Accessing Boot Log Files

On each startup the PS device records the boot process and creates a log file for it.

In the case any troubleshooting of the device is needed, these log files can be accessed through the terminal.

When connected in the terminal session, the user is located on the "user" partition of the internal flash memory. The logs are located on a different partition "log", so we first have to switch to it:

```
/ > part log
Switching to "log"
```

Here all available log files can be displayed with the "ls" command. The log files are numbered in ascending order:

```
/ > ls
d            .
d            ..
-      10849 boot.0000.log
-      12429 boot.0001.log
-      11738 boot.0002.log
-       9397 boot.0003.log
-      13135 boot.0004.log
```

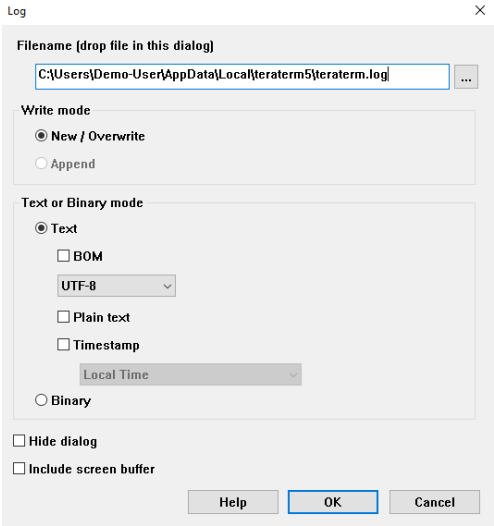
To print out a log to the terminal, the cat command can be used:

```
/ > cat boot.0004.log
>I [00:00:13] kernel.cpp::measurements_sample():2253 | Dev pc [in:0,out:0]
>I [00:00:13] heimann_pc.cpp::qstor_load():245 | Loaded Hm EE, sz=11377
>I [00:00:13] driver32x32d.cpp::ee_ram_write():240 | EE RAM store size: 11377
>I [00:00:13] qstor.cpp::segment_read():623 | Seg "hm_ee": Read 11377 bytes from FLASH
>I [00:00:13] kernel.cpp::thread_measurement():3476 | Sample now
...
```

The log output can be either copied from the terminal directly (in Tera Term in the menu ***"Edit/Copy"*** or with the shortcut ***"Alt+C"***) or if the whole terminal session is recorded, then it will be included there already.

9.3.2 Recording Live Logs

For live logging the terminal output of a device, there is the possibility to use the operation ***"File/Log…"*** in Tera Term.



Choose the location of the log file and press "OK". At this point now all output seen on the terminal will be written to this file.

For controlling the logging there is the possibility to pause it with ***"File/Pause Logging"***, or stop it completely with ***"File/Stop Logging"***.

10 Firmware Update

10.1 Direct USB connection

10.1.1 PS-250, PS-280

Following tools and files are needed to perform the firmware update:

- **esptool.exe** (the CPU manufacturer's tool to program the device)
- **bootloader.bin** (boot loader supplied by the manufacturer, once installed this later loads our program)
- **partition-table.bin** (our partition table that defines the layout of the internal data storage)
- **pikk-sense-main.bin** (firmware including our application; name may vary)
- ***.bat** (Windows batch file with the commands to control the flash process with the esptool and the three *.bin files; name may be "flash.bat", "fw.bat" or similar)

For flashing the firmware, these steps should be performed:

1. Power off device: unplug USB cable and remove any installed batteries
2. Either:
   a) ***PS-250:*** **Press and hold** the ***"SERVICE"*** button on the back of the device board (inside the case)
   b) ***PS-280:*** **Press and hold** the ***"RESET"*** button on the bottom of the device case
3. While the button is still pressed, connect the device per USB cable to the computer. This will activated the boot loader mode
4. Wait for a short moment and then **release** the pressed button (either ***"SERVICE"*** or ***"RESET"***) again
5. On the computer, make sure that no program is connected to the device per the serial connection (e.g. programs like Tera Term or PuTTY)
6. Run the provided Windows ***.bat** file, this will do following four operations:
   a) Completely erase all internal flash memory
   b) Write the boot loader,
   c) ...partition table,
   d) ...and the main application firmware to the device

7. Observe the output of the batch script in the Windows terminal and check for any errors reported by the esptool during these four operations
8. Once the script has terminated, disconnect the device from the computer

10.1.2 PS-251

For the update of the firmware only a computer and the ***.uf2** firmware file (e.g. *"ps251.uf2"*) is needed.

Perform the following steps for flashing this file to the device:

1. **Press and hold** the ***"BOOT"*** button (next to the USB port)
2. Then either:
   a) If not yet connected per USB: connect it to the computer while still holding the boot button
   b) If already connected per USB: additionally to the boot button also **press and hold** the ***"SERVICE/RESET"*** button for **10 seconds**, and then **release it again**
3. **Release** the ***"BOOT"*** button
4. Windows system should now detect a new available drive (open it in the Windows Explorer if it is not already done automatically)
5. Copy over the ***.uf2** firmware file into the directory
6. As soon as the device now detects this file here, the firmware update process is triggered. Once it is finished, the drive and also the window should close automatically, which indicates that the device was successfully rebooted after the upgrade.

10.2 Over-the-Air (OTA)

10.2.1 Providing the Firmware

The firmware binary has to be placed on a server and be accessible from the internet through either HTTP or HTTPS, the port number can be chosen freely, but has to be specified on the client side if it is not the standard port 80/443.

e.g:

- http://update.example.com/firmware/main.bin
- https://update.example.com/firmware/main.bin

10.2.2 Client Update Process

An Over-the-Air update can be started either locally or be triggered remotely via the exchange mode.

Local OTA Execution

The following three settings of the OTA module are relevant here, for more information on how to interact with the settings in the terminal, please refer to chapter "8.1.3 Commands and Terminal Interaction".

```
Module    Setting             RO   Value

OTA       RESULT              X    0
OTA       UPDATE                   0
OTA       URL                      http://update.example.com/formware/main.bin
```

First set the URL to the correct location of the firmware binary, if a custom port has to be specified, it can be added to the end of the URL with a colon (e.g.: here port 1234):

```
settings set ota url http://update.example.com/firmware/main.bin
settings set ota url http://update.example.com/firmware/main.bin:1234
```

Then write a "1" into the UPDATE variable to trigger the update process:

```
settings set ota update 1
```

The device will now reboot and try to download the firmware from the server and perform the update.

If the update was successful, then the device will store a "1" in the read-only variable RESULT:

```
Module    Setting             RO   Value

OTA       RESULT              X    1
```

If any error was encountered during the update process, the error code (>1) will be stored here as the RESULT instead:

```
General error:
    10    Protocol error
    11    URI/URL error
    12    System error
    13    Response error
    14    Parsing error

Transport error:
    20    Network error
    21    Remote error

OTA error:
    30    OTA init error
```

```
    31    OTA write error
    32    OTA validation error
```

Please note, that the RESULT value is not stored permanently here, but rather will be sent out with the following MQTT data message and then resets itself to "0" again afterwards.

In the message the RESULT is transferred with the "ota" key::

```
{
  "serial" : "PS280-123456",
  …
  "ota" : 1
}
```

Remote Triggering via Exchange Mode

OTA updates can also be done via the exchange mode described in the chapter "8.3 Remote Configuration via Exchange Mode ".

For this, a message can be sent down to the device once its exchange mode is active to both set the URL and also trigger the update process itself:

```
{
  "OTA": [
    {
      "name": "URL",
      "value": "http://update.example.com/firmware/main.bin:1234"
    },
    {
      "name": "UPDATE",
      "value": 1
    }
  ]
}
```

The device will then notify on the upstream topic that the commands have been processed with the result message:

```
{
  "ret": 0
}
```

After the device has performed the installation, it will provide the result of the OTA update in its next data package, same as for the local execution:

```
{
  "serial": "PS280-123456",
  …
  "exch": 1,
  "ota": 1
}
```

Same as before, the "ota" will show either a "1" for a successful installation, but if the update attempt was unsuccessful, the corresponding error code will be shown here.

11 Installation Notes

[TBD]

11.1 PS-250 + PS251

When connecting any number of PS-251 extenders to the PS-250, consider following points regarding the cable lengths between the devices.

Viewing the setup starting at the PS-250:

- Each PS-251 uses up its necessary current, device per device
- Keep the first cables as short as possible, as the most power is running on these parts (highest current on cable between PS-250 and first PS-251, maximum voltage drop on the whole system here)
- Cables to the more remote PS-251 towards the end of the device chain can be longer
- Generally ~30m of cable in total should not cause any problems

11.2 PS-280

[TBD]

12 Troubleshooting

12.1 PS-250 LED Signaling

LED Signal	Interpretation
Single GREEN fade in+out	Cold boot of the device when plugging device into a power source
Single or multiple BLUE fade in+out	Registration in process (LED duration depending on completion time
Repeating RED fade in+out	Registration of the device failed
Single WHITE fade in+out	A button press command was successfully recognized and is processed
Single BLUE fade in+out	Shutdown of the device is complete after a shutdown command was triggered by the user