ThetaMon3 Command-Line Reference

Refer to file “Interpreter.cpp”

Connect your computer to the ESP32 via USB-cable. Start a terminal-application like Gtk-Term, set the right port, i.e. /dev/ttyUSB0 or /dev/ttyACM0. Turn local echo off.
You could also use Visual Studio Codes Serial Monitor. Push the “Toggle Terminal Mode” button for enabling typing in commands.
Remember to shut down the terminal, or disconnect it, if you want to flash the ESP, or write the file-sytem image.

String arguments must be in double quotes (i.e. “Workshop”), Floats have to be written with a dot as decimal separator.

Like in Unix-terminals, you could access the history with arrow up / down.

```
---------------------------------
setSensId <Sensor-Hash> <min> <max> <sensor-type> <relay-channel> “<shortname>”

float <min>:
      Lower threshold, where the relay-channel is switched on.
float <max>:
      Upper threshold, where the relay-channel is switched off,
int <sensor-type>:
      0 – Temperature
      1 –  Humidity
      2 –  Pressure
      3 –  Relay (We treat Relays as sensor, to read their states)
int <relay-channel>:
      0 – No channel. This sensor doesn’t switch anything.
      1 – Channel 1.
      2 – Obviously, Channel 2.
string “<shortname>”:
      Name of the sensor. Because of history-reasons, it must be exactly eight
      bytes long. I recommend to use underscores for padding, if your name is
      shorter.

example:
      setSensId 3822322055 10.0 12.5 0 1 "Test 007"
answer:
      SensId ignored: 3822322055 (87:01:d4:e3:00:00:00:00)
      ** DONE **

---------------------------------
calcHash <newCommand>

Calculates the hash value of the argument. Used only, if new commands will be
implemented in code.

example:
      calcHash blahdieblubb
answer:
      Hash = 3496999634

---------------------------------
```

getSensIdTable

Prints the content of the file „ID_Table_U64.txt“, that is stored in LittleFs filesystem.

example:
      getSensIdTable
answer:
      setSensId 288253003812035532 -12.5 -12.7 0 0 "WohnzTmp"
      setSensId 360310597849963468 40.0 80.0 1 0 "WohnzHum"
      setSensId 432368191887891404 846.0 1100.0 2 0 "WohnzPrs"
      setSensId 720598568039603148 0.0 1.0 3 0 "WohnzRl1"

      ** DONE **

---
getStationId

Prints the base-address, used for sensor-id generation. It is the Mac-address of the device, added two zero bytes. Was used in ThetaMon2.

example:
      getStationId
answer:
      Station ID = 149609863437348 (24:dc:c3:c2:11:88:00:00)

---
printMeasures

One of the important commands. Prints the measurement of all connected sensors.

For every sensor a block, like following is reported:

```
sensId = 7638694032459628328 (28:ff:a6:40:c0:17:02:6a)
shortname = Test 005    sensType = TEMP
meanValue = 28.55 lastUpdateTick = 1213
valueIndex = 2          values = 28.50 28.50 28.69 28.56 28.50
minVal = -12.50         maxVal = -12.70
sensChan = CH_2         relayNr = 0
configChanged = false   isTimeout = false
```

sensId: The address of the sensor as hash and 8-byte value
shortname: name of the sensor-address
sensType: type of the sensor
meanValue: The average of the internal array. Every minute a measuring is done, every five minutes the meanValue is reported to the Mqtt-broker.
values: The internal values, from which meanValue is built.
minVal / maxVal: Thresholds, where relay is switched on/off.
sensChan: Internal. Shows to which channel the sensor is connected to.
relayNr: The relay, this sensor is bound to.
configChanged: Internal. Flags, if after booting the setting of the sensor was changed. After saving the config (see saveSensIdTable), this flag is cleared.
isTimeout: If there is no answer, longer than MEASUREMENT_TIMEOUT_SEC (Config.h) the sensor is flagged timeout and is no more reported to the broker.

```
listDir

Shows the contents of LittleFs. Usually there should only be two files,
ID_Table_U64.txt and  MqttConfig.txt.

example:
      listDir

answer:
      Listing directory: /
      FILE: ID_Table_U64.txt  SIZE: 2264
      FILE: MqttConfig.txt    SIZE: 134

---------------------------------
readFile “filename”

Prints the contents of a file. The filename must be preceded by “/”!

example:
      readFile "/MqttConfig.txt"

answer:
      setMqttHost 192 198 153 122
      setMqttPort 1883
      setMqttSpot "Testboard"

      startWifi "My-SSID" "08151234567Passkey"

---------------------------------
saveSensIdTable

Saves any changes, made to the sensor-config to the LittleFs file-system.
(At time of writing this, the command fails and the device reboots, if no
changes are made to the sensor-table before.)

example:
      saveSensIdTable

answer:
      ** DONE **

---------------------------------
getMacAddress

Prints the Mac-Address of your device.

example:
      getMacAddress

answer:
      DEC 36:220:195:194:17:136 (HEX 24:dc:c3:c2:11:88)

      ** DONE **

---------------------------------
```

```
setMqttHost <Ip-Address_of_your_Mqtt-broker>

Set the Mqtt-Server address. Must be written with blanks, instead of dots
between the numbers.

example:
      setMqttHost 192 158 234 136

answer:
      ** DONE **

---------------------------------
setMqttPort

Mqtt-Server-port. Usually 1883.

example:
      setMqttPort 1883

answer:
      ** DONE **

---------------------------------
startWifi "<SSID>" "<PASS>"

Fires up the Wifi with given credentials.

example:
      startWifi "someSSID" "somePass"

---------------------------------
stopWifi

Shuts the Wifi down.

---------------------------------
setMqttSpot

Sets the name of the location. This name is used in the Mqtt-publishing.

example:
      setMqttSpot "someLocation"

answer:
      ** DONE **

---------------------------------
```

getMqttConf

Prints the current Mqtt-Setup.

example:
      getMqttConf

answer:
      WIFI is connected, local IP: 192.168.178.98
      MqttHost: 192.238.142.127:1883
      Mqtt publish sensordata: 'someLocation/sens'
      Mqtt publish logs: 'someLocation/log'
      Mqtt subscribe commands: 'someLocation/cmd'

      ** DONE **

---
storeWifi

Stores the current Wifi settings, to “MqttConfig.txt”. The file will be appended with a startWifi command.

---
storeMqttSpot

Appends “MqttConfig.txt” with a setMqttSpot command. The current location is used as argument.

---
storeMqttHost

Appends “MqttConfig.txt” with setMqttHost command. The current mqtt-host setting is used.

---
storeMqttPort

Same with setMqttPort.

---
tstRelay <Relay-Nr> <State>

Switches the corresponding relay on or off. A test-timer is started, currently 2 min, that preserves the state of the relay beeing changed from a sensor reaching a threshold. After the timer is expired, the relay is automatically switched off. If any sensor is within its thresholds, the relay will be switched in the next cycle.

int <Relay-Nr>: 1 or 2, no other values allowed.
int <State>: 0 or 1, no other values allowed.

example:
      tstRelay 1 1

answer:
      ** DONE **

---

cycle

A cycle is done every minute. In a cycle all sensors are read and the relay-states are calculated. In test or setup situations, you don't necessarily want to wait that long. This command triggers cycling.
After hitting enter, the ESP seems to be frozen for a second or two. Don’t be nervous, just wait.
After five cycles the mqtt reporting is done.

example:
      cycle

answer:
      ** DONE **

---
shutup

This command inhibits any output to the terminal. It is useful, when you run some scripts on your computer, that push messages right into the serial.

example:
      shutup

answer:
      (nothing happens)

---
talk

Enables the terminal outputs again.

example:
      talk
answer:
      Serial print is active.

      ** DONE **

---
reboot

Reboots the device.

---
exit

On ESP devices, it does the same as reboot.

---
clear

Clears the screen.
