# SENSORhub SEH

Revision A | 2026-07-23

## Overview

The SENSORhub SEH is a controller ("hub") that drives addressable LED strips and connected SMC buttons (SENSORswitches) for guidance and pick-by-light applications.

The hub supports up to 5 LED strips (LED_STRIP_1 through LED_STRIP_5). Each strip can contain multiple segments, and each segment can be individually configured with a start/stop position, an effect, a speed and up to three colors.

See Hardware Assembly for wiring and power supply, and Connectivity for network setup, status signalisation and fallback behavior.

## Control Interfaces

The SENSORhub SEH can be controlled and monitored through two interfaces that operate in parallel — the most recently received command determines the current state:

- **MQTT Communication** — device onboarding, LED strip configuration and activation, health/status reporting, firmware updates and time distribution.
- **Modbus Interface** — Modbus TCP control of the LED strips and SMC buttons, plus read-only status registers.

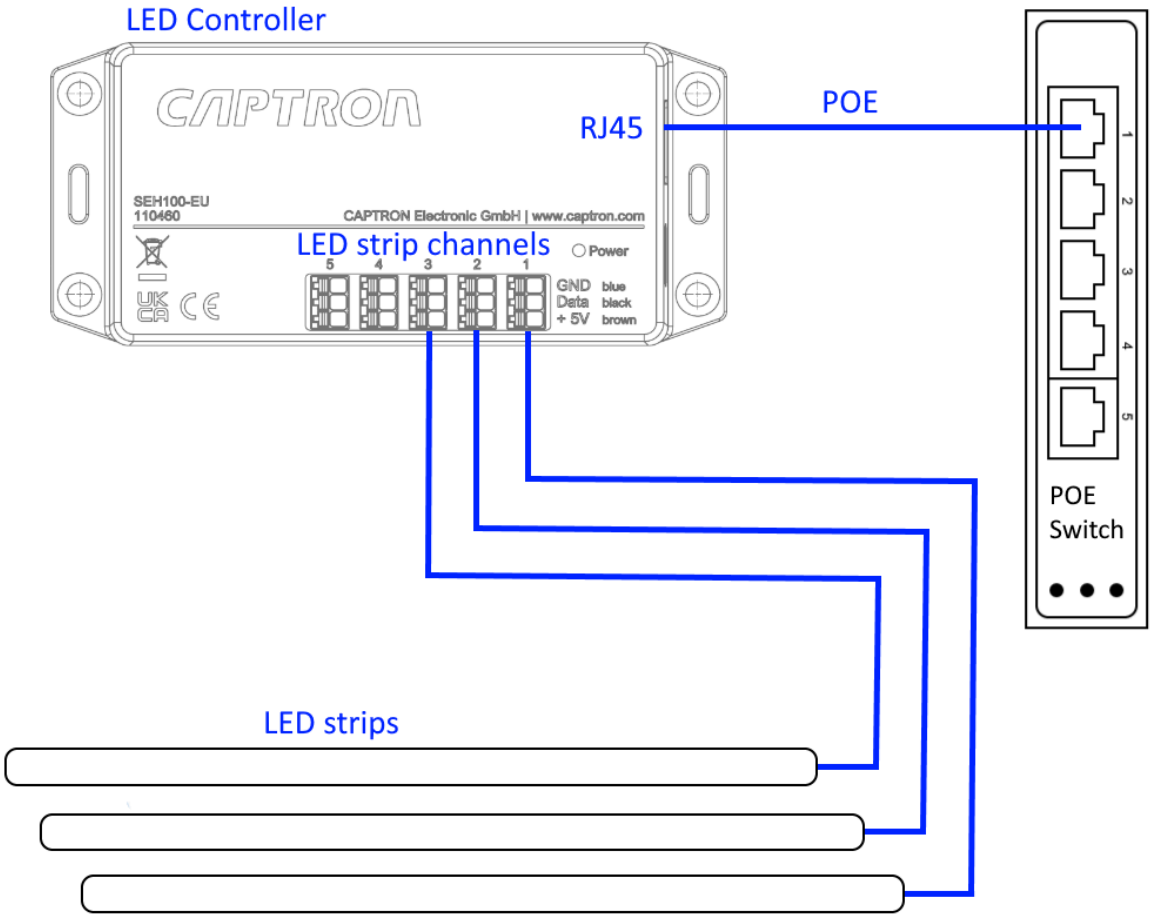
## Device Variants

| Device | Description |
|---|---|
| SEH11 | SENSORhub controller for SMC buttons (SENSORswitches) only — no LED strip support |
| SEH101 | SENSORhub controller for up to 5 addressable LED strips, with output diagnosis (wiring-fault detection on the output clamps) |
| SEH201 | SENSORhub controller for up to 5 addressable LED strips and SMC buttons (SENSORswitches), with output diagnosis |

> **NOTE**
>
> Some features are device-dependent. LED strips (up to 5) are supported on SEH101 and SEH201; SMC buttons are supported on SEH11 and SEH201. Output diagnosis is available on SEH101 and SEH201; diagnosis messages are reported on SEHx01 devices. See the individual interface pages for details.

# Hardware Assembly

The cable connections between all parts of the system are shown in the schematic below.




Each LED strip connects to one of the LED strip channels (outputs 1–5). The strip conductors are: `GND` (blue), `Data` (black) and `+5V` (brown).

# Connectivity

This page describes how a SENSORhub SEH device connects to the network, how its connection state is signaled, how it is configured offline through the web setup page, and how it falls back when the configured connection cannot be established.

## Status Signalisation

Applies to SEH101 and SEH201.

At startup the device status is signaled by the connected LED strips and by the status LED (SEHx01). To help during installation (mapping strips to outputs), a different count of LEDs lights up on each output: one LED on output 1, two LEDs on output 2, and so on.

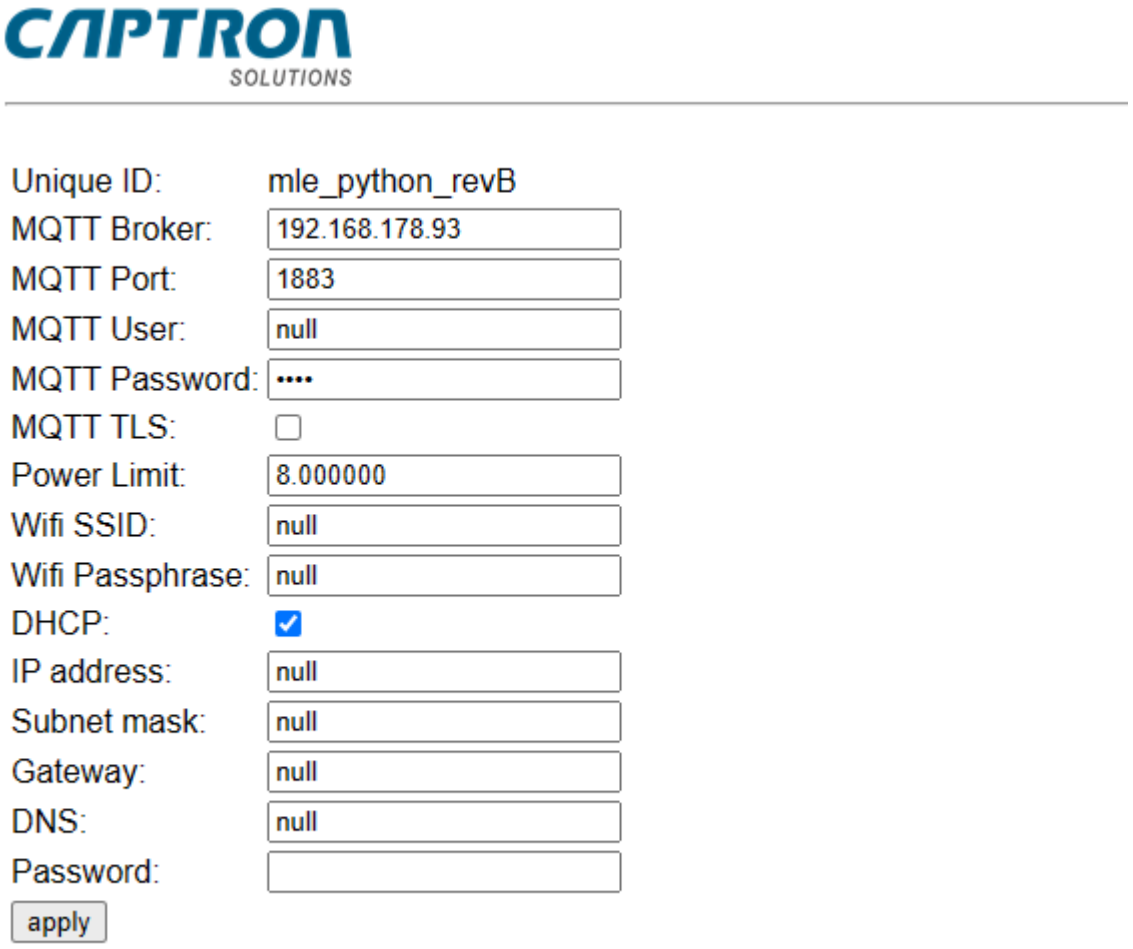
The signalisation colors are shown until the first Set/Data command is received.

| Color | Status |
|---|---|
| Red | Ethernet/WiFi connection attempt, no Ethernet/WiFi connection. |
| Yellow | MQTT connection attempt, no MQTT connection. |
| Blue | Fallback mechanism active (planned from V0.2.33/V0.3.6), check configuration and/or network connection. |
| Cyan/Green-Blue | Connection established — to the onboarding-service (if no user MQTT is configured), or to the user MQTT (if it is configured). |

## Web Setup Page

Applies to SEH101 and SEH201.

As an option for offline configuration, the integrated setup page can be used. It can be accessed from any browser by entering `http://` followed by the IP address of the device. By default the device tries to obtain its IP address from a DHCP server; the assigned address can be looked up in that server/router. The web setup page is available from firmware version V0.2.27-29.




To apply settings, enter `pythonhead22` in the Password field (last line) and confirm with the apply button.

> **NOTE**
>
> For unused entries (for example MQTT authentication or WiFi), set the field content to `null`.

## Static IP Setting

If a static IP setting is required (for example, when no DHCP server can be used), fill in at least the three fields: IP address, Subnet mask and Gateway. If there is no gateway, enter a valid address here anyway (for example the device's own address).

From firmware version V0.2.27-57, the DNS field can be left `null`. The device will then omit trying to connect to the onboarding-service.

## Certificates for User MQTT Connection

If TLS is to be used together with client certificates (mTLS), the web setup page also offers the possibility to upload certificate files (for example `crt` / `pem` / `key`). This requires at least firmware

version V0.3.7. The file name and extension have no relevance. Make sure that the TLS checkbox is set.

Also make sure to upload the proper issuer certificate as the CA cert, so that the TLS engine can verify the device certificate as well as the incoming server certificate. Self-signed certificates can also be used.

If the file is uploaded successfully, "file loaded" appears. To overwrite a file, choose a new file and select "Upload". All files can be removed with "Remove". After uploading all necessary files, restart the device by clicking "Restart".

# Fallback Functionality

Applies to SEH101 and SEH201.

## Priority

From firmware version V0.2.27-41, the Ethernet port has priority over WiFi. If a connection can be established through LAN, the WiFi setting is ignored. If no Ethernet connection can be established and no WiFi is configured, the device tries to connect to the fallback WiFi (EU variants only) — this takes up to 45 seconds.

## WiFi Fallback

If a configured WiFi network cannot be reached, the device tries to access this hard-coded WiFi:

- SSID `SEH_fallback_wifi`
- Passphrase `fallback_pw`

This fallback WiFi network can be created, for example, by setting up a hotspot on a mobile phone with the specific name and passphrase.

Once the connection to the fallback WiFi is established, the device connects to the onboarding-service and gets the latest configuration. In this way, a faulty WiFi configuration can be corrected.

> **NOTE**
>
> The WiFi functionality is only available on EU variants. On other variants such as JP, CN or US, the WiFi module is disabled.

## Static IP Fallback

If a device with a static IP setting cannot access the network after about 45 seconds, it falls back and tries to obtain its network settings from a DHCP server.

If the fallback is successful, the device connects to the onboarding-service and gets the latest configuration. A faulty IP setting can be corrected this way.

## DHCP Fallback (APIPA)

If the device is configured to obtain an address from a DHCP server and this server is not reachable, the device assigns itself this network setting (within the APIPA range) after about 45 seconds:

- IP address: `169.254.101.201`
- Subnet: `255.255.0.0`

With a direct Ethernet connection to a computer, it can be reconfigured using the web setup page. This fallback method is available from firmware version V0.2.27-63.

## Three-Step Fallback

From firmware version V0.3.7-1, the fallback mechanisms are simplified and unified.

Independent of the configuration (static IP or DHCP), the fallback steps are always:

Static IP → DHCP → APIPA → then start again at the beginning. Each step is active for about 45 seconds.

# MQTT Communication

SENSORhub SEH devices are controlled and monitored over MQTT. This page describes the MQTT topics and their message payloads. A Modbus TCP interface is available in parallel — see Modbus Interface.

## Definitions

- `{product}` is the name of the product, e.g. `SEH101` or `SEH201`.
- `{device-id}` is the unique name of the device and consists of a randomly generated word combination such as `GroovySquareTongue` or `TinyHappyGroup`. This id is used to build the topics and is printed on the device.
- `{Content definition}` is the part of the topic after the `{device-id}`, e.g. `/Pub/MAM`.

All topics follow this pattern:

```
captron.com/{product}/nd/{device-id}/{Content definition}
```

## Device Information

### Publish — `/Pub/MAM`

Direction: device publishes.

```
{
  "Content": "/Pub/MAM",
  "BoardName": "lucky_python",
  "Manufacturer": "CAPTRON",
  "Model": "PYTHON-Head",
  "ProductCode": "123456789",
  "SoftwareVersion": "v0.0.1"
}
```

The device broadcasts data which is generic and individual to a device, the configuration which is given to the device during the onboarding process, and all methods which can be called.

### Subscribe — `/Get/MAM`

Direction: device subscribes. Payload: none.

Request device information. The device replies on the corresponding `/Pub/MAM` topic.

# Configuration

## Subscribe — `/Set/Config/MQTT`

Direction: device subscribes.

```
{
  "Content": "/Set/Config/MQTT",
  "Vendor": "captron",
  "Port": 1883,
  "MQTTServer": "update.oneGrid.captron.com",
  "MQTTUsername": "Captron2022",
  "MQTTPassword": "qwxYT321!",
  "Encryption": false
}
```

Provide connection information specific to the owner of the application, such as the server configuration. If no login credentials are needed, use `null` as the value of `MQTTUsername` and `MQTTPassword`.

This message is used during the automatic onboarding process. Be aware that after a power cycle the device will request this information from the onboarding system again.

## Subscribe — `/Set/Config/LedStrip`

Direction: device subscribes.

```
{
    "Content": "/Set/Config/LedStrip",
    "Demo": false,
    "LED_STRIP_1":{
        "ColorAlignment": {"Active": true, "R": 255, "G": 180, "B": 255}
    },
    "LED_STRIP_2":{
        "ColorAlignment": {"Active": true, "R": 255, "G": 180, "B": 255}
    },
    etc...
}
```

Use-case specific configuration of the LED strips (e.g. length as count of LEDs).

- For FW v0.2.10 and higher this message is no longer mandatory. The default used is 180 LEDs per strip.
- The boolean `Demo` enables/disables the demo mode. If enabled, the device plays different colors and effects on output 1 and 2. This setting is stored persistently in memory — if you enable the demo mode it is also active after a power cycle.
- From FW V0.3.10-13-G225F748 onward, color alignment is possible for each LED strip. Set the `Active` flag to `true` and set the corresponding RGB values in the range 0–255.

# Activate LED strip

## Subscribe — `/Set/Data/LedStrip`

Direction: device subscribes.

```
{
  "Content": "/Set/Data/LedStrip",
  "LED_STRIP_1": {
    "Active": true,
    "Segments": [
      {
        "StartLED": 0,
        "StopLED": 30,
        "Speed": 190,
        "Effect": 1,
        "Colors": [
          {
            "R": 0,
            "G": 150,
            "B": 0
          },
          {
            "R": 0,
            "G": 150,
            "B": 0
          }
        ]
      }
    ]
  }
}
```

Light up the LEDs.

- Use the `Active` flag to switch the strip on or off.
- Effects:
  - 1: FX_MODE_STATIC
  - 2: FX_MODE_BLINK
  - 3: FX_MODE_FLASH
  - 4: FX_MODE_LEADING_LINES
  - 5: FX_MODE_LEADING_LINES_REVERSE
  - 6: FX_MODE_MULTIPICKER
  - 7: FX_MODE_PICK (needs min. FW version V0.3.7)
  - 8: FX_MODE_PLACE (needs min. FW version V0.3.7)
  - 200: FX_MODE_CUSTOM_KNIGHT_RIDER
- Speed from 1 (slow) to 250 (fast).
- Multiple colors apply only to some effects such as FX_MODE_MULTIPICKER. In that case the colors change between the RGB values specified in the `Colors` array. For all other effects only the first color in the array applies.

- 6 segments with different colors/effects can be set. Depending on the maximum MQTT message size, more segments can be set. The MQTT message size must be below 4k.
- This command can be ignored if a power limit is reached. In this case an error message is logged to inform the user.

## Control of SMC buttons (SEH20x/SEH11 devices)

Control of SMC buttons (SENSORswitches) is available on SEH11 and SEH201. The SMC buttons connected to the hub are controlled through a single topic. The Modbus interface offers the same SMC control in parallel — see Modbus Interface.

Implemented in FW V0.2.5.

### Subscribe — `/Set/Data/Smc`

Direction: device subscribes.

```
{
  "Content": "/Set/Data/Smc",
  "Address": "{Button address, HUB or ALL_SENSORS}",
  "CommandType": "{Command Type}",
  "Offset": "{Offset}",
  "Payload": "{Payload}"
}
```

- **Address** — a specific button address, `HUB` (the SEH itself), or `ALL_SENSORS` (broadcast to all buttons).
- **CommandType** and **Offset** together select which command is executed and how the `Payload` string is interpreted.

The following commands are available:

| Command | CommandType | Offset | Payload |
|---|---|---|---|
| Set LED ring and display | SET_PARAMETER | PRE_BUTTON_MODE or POST_BUTTON_MODE | `{ENABLED, DISABLED or LONGPRESS ENABLED}/{Quantity}/{LED ring color}/{LED ring effect}/{Display text}` |
| Set button rotation | SET_PARAMETER | BUTTON_ROTATE | `BUTTON_ROTATION_NORMAL` or `BUTTON_ROTATION_UPSIDEDOWN` |
| Set longpress time | SET_PARAMETER | LONGPRESS_TIME | Milliseconds divided by 10 (max. 2.55 seconds) |
| Assign button address | SET_STATUS | ADDRESS_TO_BE_ASSIGNED | Number to assign |
| Special commands (switch Pre/Post state, reboot, etc.) | SET_STATUS | SPECIAL_COMMANDS | `PRE_PRESSED_STATE`, `POST_PRESSED_STATE`, `REBOOT`, `SHOW_CURRENT_ADDRESS` or `LONG_PRESSED_STATE` |
| Polling function (Address = HUB) | SET_PARAMETER | (none) | Polling addresses separated by `/` (up to 6), e.g. `1002/1003/1004`; `0000` switches polling off |

**Special commands** (Offset `SPECIAL_COMMANDS`):

- `PRE_PRESSED_STATE` — set to pre state
- `POST_PRESSED_STATE` — set to post state
- `REBOOT` — restart button
- `SHOW_CURRENT_ADDRESS` — show button address
- `LONG_PRESSED_STATE` — set to longpress state (needs min. SEH firmware V0.2.27-3)

> **NOTE**
>
> Older firmware versions do not apply the **button rotation** setting after a power cycle. In that case the command must be resent after each power cycle of the SMC button.
> The **longpress time** command needs min. firmware V5.3.9 on the SMC button.

## Publish — `/Pub/Data/Smc`

Direction: device publishes.

When polling is active for a button address, a button touch is reported on this topic:

```
{
  "Content": "/Pub/Data/Smc",
  "Address": "{touched button address}",
  "CommandType": "SET_STATUS",
  "Payload": "POST_PRESSED_STATE"
}
```

## Setting LED ring and display

For the **Set LED ring and display** command (CommandType `SET_PARAMETER`, Offset `PRE_BUTTON_MODE` or `POST_BUTTON_MODE`), the `Payload` is a `/`-separated string:

```
{ENABLED, DISABLED or LONGPRESS_ENABLED}/{Quantity}/{LED ring color}/{LED ring effect}/{Display text}
```

**Button mode** (first field):

- `ENABLED` — button can be confirmed; LED ring and display can be controlled.
- `DISABLED` — button can NOT be confirmed; LED ring and display can be controlled.
- `LONGPRESS_ENABLED` — button goes to the longpress state after keeping the touch for a certain time (default 2 seconds); the LED ring and display for that state are set using the `POST_BUTTON_MODE` offset.

**Quantity** — a number from 1 to 9999. It is automatically aligned in the middle of the display.

**LED ring color:**

- `COLOFF`
- `COLRED`
- `COLGREEN`
- `COLBLUE`
- `COLYELLOW`
- `COLMAGENTA`
- `COLCYAN`
- `COLWHITE`

**LED ring effect:**

- `SOLID_RING`
- `FLASH_RING`
- `CONFIRM`
- `POINT_ANIMATED_CLOCK`
- `CIRCLE_ANIMATED_CLOCK`
- `SOLID_ARROW_UP`, `SOLID_ARROW_DOWN`, `SOLID_ARROW_LEFT`, `SOLID_ARROW_RIGHT`
- `FLASH_ARROW_UP`, `FLASH_ARROW_DOWN`, `FLASH_ARROW_LEFT`, `FLASH_ARROW_RIGHT`
- `ANIMATED_ARROW_UP`, `ANIMATED_ARROW_DOWN`, `ANIMATED_ARROW_LEFT`, `ANIMATED_ARROW_RIGHT`
- Diagonal arrows: `SOLID_ARROW_UP_LEFT`, `SOLID_ARROW_UP_RIGHT`, `SOLID_ARROW_DOWN_LEFT`, `SOLID_ARROW_DOWN_RIGHT`, `FLASH_ARROW_UP_LEFT`, `FLASH_ARROW_UP_RIGHT`, `FLASH_ARROW_DOWN_LEFT`, `FLASH_ARROW_DOWN_RIGHT`

> **NOTE**
>
> The diagonal arrow effects need min. firmware V5.3.7 on the SMC button.

**Display text** — max. 4 characters. Some characters cannot be displayed due to 7-segment restrictions. Use `@` as a space character (examples: `@Go@`, `donE`).

If both **Quantity** and **Display text** are defined, **Quantity** is used. When **Display text** is used, no automatic alignment is done — the user must take care of it (e.g. a left-aligned number as `42@@`, a right-aligned number as `@@42`). If **Quantity** is used, the alignment is done automatically.

## Examples

Sending to the SMC button with address `1000`.

Set LED ring color to green (flashing) and display a quantity of `42`:

```
{
  "Content": "/Set/Data/Smc",
  "Address": "1000",
  "CommandType": "SET_PARAMETER",
  "Offset": "PRE_BUTTON_MODE",
  "Payload": "ENABLED/42/COLGREEN/FLASH_RING"
}
```

Set LED ring color to blue and display `donE` in the post-pressed / longpress state:

```
{
  "Content": "/Set/Data/Smc",
  "Address": "1000",
  "CommandType": "SET_PARAMETER",
  "Offset": "POST_BUTTON_MODE",
  "Payload": "ENABLED//COLBLUE/SOLID_RING/donE"
}
```

Switch the button to the pre-pressed state:

```
{
  "Content": "/Set/Data/Smc",
  "Address": "1000",
  "CommandType": "SET_STATUS",
  "Offset": "SPECIAL_COMMANDS",
  "Payload": "PRE_PRESSED_STATE"
}
```

Set the longpress time to 1 second:

```
{
  "Content": "/Set/Data/Smc",
  "Address": "1000",
  "CommandType": "SET_PARAMETER",
  "Offset": "LONGPRESS_TIME",
  "Payload": "100"
}
```

Sending to all connected SMC buttons — switch all buttons to the pre-pressed state:

```
{
  "Content": "/Set/Data/Smc",
  "Address": "ALL_SENSORS",
  "CommandType": "SET_STATUS",
  "Offset": "SPECIAL_COMMANDS",
  "Payload": "PRE_PRESSED_STATE"
}
```

## Health and Status

### Publish — `/Pub/Health/LedStrip`

Direction: device publishes.

```
{
  "Content": "/Pub/Health/LedStrip",
  "Firmware": "V0.1.0",
  "LED_STRIP_1": {
    "Length": 42,
    "Active": true,
    "CheckPassed": true
  },
  "LED_STRIP_2": {
    "Length": 42,
    "Active": false,
    "CheckPassed": true
  },
  "LED_STRIP_3": {
    "Length": 42,
    "Active": false,
    "CheckPassed": true
  },
  "LED_STRIP_4": {
    "Length": 0,
    "Active": false,
    "CheckPassed": true
  },
  "LED_STRIP_5": {
    "Length": 0,
    "Active": false,
    "CheckPassed": true
  }
}
```

The health of the LED strips connected to the board. This message is published after powering on the device.

## Subscribe — `/Get/Health/LedStrip`

Direction: device subscribes. Payload: none.

Request LED strip health and information. The device replies on the corresponding `/Pub/Health/LedStrip` topic.

## Publish — `/Pub/Health/SysLog`

Direction: device publishes.

Payload: event logging (info, warn, error, etc.), version, uptime, IP address, etc.

The syslog of a device. This message is published every 6 h and also when the LED strip health is requested. Diagnosis messages (e.g. detected wiring errors, overloads, etc.) are also published on this topic — these messages are marked with `[DIAGNOSIS]`.

# Common settings

## Subscribe — `/Set/Config/Common`

Direction: device subscribes.

```
{
    "Content": "/Set/Config/Common",
    "LogLevel": {1 to 6},
    "StartDiagnosis": true
}
```

**LogLevel** — sets the log level. The default setting is `3` (Warning). The following levels can be set:

- 1 – Fatal
- 2 – Error
- 3 – Warning (default)
- 4 – Notice (logs e.g. the response when an LED segment is activated)
- 5 – Info
- 6 – Trace

Log messages with the chosen and lower numbers (higher severity) are published on the syslog topic. E.g. setting the log level to 4 forces the device to publish log messages with the rankings Fatal, Error, Warning and Notice. The log level setting is not saved in flash — the device always starts with the default setting.

**StartDiagnosis** (only for devices SEH101 and SEH201) — starts diagnosis on the output clamps, checking for e.g. wiring faults. Possible errors are published on the syslog topic. Diagnosis is automatically run at startup; this flag triggers the functionality once.

# Firmware Update

## Subscribe — `/Call/FirmwareUpdate`

Direction: device subscribes.

```
{
  "Content": "/Call/FirmwareUpdate",
  "Url": "http://onegrid-
fwupd.germanywestcentral.cloudapp.azure.com:80/ota/firmware_V0.3.9-pythonhB.img"
}
```

Start the update process.

> **NOTE**
>
> SEH201 uses firmware files named `firmware_V0.x.y-pythonhB_smc.img`.

# Time distribution

## Subscribe — `/Set/Config/Time`

Direction: device subscribes. Topics:

```
captron.com/Time
captron.com/{product}/nd/{device-id}/Set/Config/Time
```

```
{
  "Content": "/Set/Config/Time",
  "DateTime": "%Y-%m-%d %H:%M:%S"
}
```

Distribute time to each device. String with the format `%Y-%m-%d %H:%M:%S`, e.g. `2023-02-28T11:18:30`.

# Status signalisation

At startup the device status is signaled by the connected LED strips. To help during installation (mapping strips to outputs), a different number of LEDs light up on each output: one LED lights up on output 1, two LEDs light up on output 2, etc. The signalisation colors are shown until the first `/Set/Data` command.

| Color | Status |
|---|---|
| Red | Ethernet/WiFi connection attempt, no Ethernet/WiFi connection |
| Yellow | MQTT connection attempt, no MQTT connection |
| Blue | Fallback mechanism active (planned from V0.2.33/V0.3.6), check configuration and/or network connection |
| Cyan/Green-Blue | Connection established — to the onboarding service (if no user MQTT is configured) or to the user MQTT (if it is configured) |

# Diagnosis messages (for SEHx01 devices)



Diagnosis messages are published on the syslog topic.

| Failure | Message |
| --- | --- |
| Voltage for LEDs not in range (4.5 V – 5.5 V), e.g. weak USB power supply | `[DIAGNOSIS] led voltage outside range` |
| LED current below expected current for activated segments, e.g. LED strip damaged, LEDs activated outside of strip length | `[DIAGNOSIS] expected current not in range` |
| Wiring error or over current, details in ERROR_BITS:<br>• 1: wiring error on Data/GND pin<br>• 2, 4: wiring error on Vled/GND pin<br>• 32: LED power reached power supply limit | `[DIAGNOSIS] failed: {ERROR_BITS}` |

# Modbus Interface

## Overview

This document describes the Modbus TCP/IP interface for controlling the LED strips and SMC buttons of SEHx01/SEH11 devices. The Modbus interface operates in parallel with the existing MQTT control. Both interfaces can be used simultaneously.

## Physical Interface

| Parameter | Value |
|---|---|
| Protocol | Modbus |
| Interface | TCP/IP |
| Device ID | 1 (configurable) |

## Register Overview

### Holding Registers (Function 03/06/16)

The controller supports 5 LED strips (LED_STRIP_1 through LED_STRIP_5). Each strip can contain multiple segments. Up to 8 segments per strip are configurable. Each segment supports up to 3 colors.

### Global Registers

| Register | Description | Data Type | Range | R/W |
|---|---|---|---|---|
| 0 | Modbus Device ID | UINT16 | 1-247 | R/W |
| 2 | Apply/Trigger | UINT16 | 0/1 | W |
| 3 | Reserved | - | - | - |
| 4 | Reserved | - | - | - |

**Apply/Trigger (Register 2):** After writing all desired strip/segment registers, register 2 must be set to `1` to apply the changes. The register is automatically reset to `0` after the changes are applied.

## Strip Registers

Each strip occupies a block of 100 registers. The base address for strip N (N = 1..5) is calculated as:

**Base Address = 100 * N**

| Offset | Description | Data Type | Range | R/W |
|---|---|---|---|---|
| 0 | Active | UINT16 | 0/1 | R/W |
| 1 | Segment Count | UINT16 | 0-8 | R/W |
| 2-9 | Reserved | - | - | - |

## Segment Registers

Within a strip block, each segment occupies 11 registers. The base address for segment M (M = 0..7) within strip N is:

**Segment Base Address = (100 _ N) + 10 + (M _ 11)**

| Offset | Description | Data Type | Range | R/W |
|---|---|---|---|---|
| 0 | StartLED | UINT16 | 0-311 | R/W |
| 1 | StopLED | UINT16 | 0-311 | R/W |
| 2 | Effect | UINT16 | see Effects | R/W |
| 3 | Speed | UINT16 | 1-246 | R/W |
| 4 | Color 1 - Red | UINT16 | 0-255 | R/W |
| 5 | Color 1 - Green | UINT16 | 0-255 | R/W |
| 6 | Color 1 - Blue | UINT16 | 0-255 | R/W |
| 7 | Color 2 - Red | UINT16 | 0-255 | R/W |
| 8 | Color 2 - Green | UINT16 | 0-255 | R/W |
| 9 | Color 2 - Blue | UINT16 | 0-255 | R/W |
| 10 | Color 3 - Red | UINT16 | 0-255 | R/W |
| 11 | Color 3 - Green | UINT16 | 0-255 | R/W |
| 12 | Color 3 - Blue | UINT16 | 0-255 | R/W |

## SMC Button Control (Holding Registers)

In addition to the LED strips, the connected SMC buttons can also be controlled via Modbus. This corresponds to the existing MQTT control (topic `/Set/Data/Smc`). Both paths internally generate the same SMC command and can be used in parallel.

The SMC block occupies a dedicated register range starting at base address **1000** and has its **own Apply trigger**, independent of the LED Apply (register 2).

| Register | Description | Data Type | Range | R/W |
|---|---|---|---|---|
| 1000 | SMC Address | UINT16 | see below | R/W |
| 1001 | SMC CommandType | UINT16 | 0-2 (see below) | R/W |
| 1002 | SMC Offset | UINT16 | 0-5 (see below) | R/W |
| 1003 | Button Mode | UINT16 | 0-2 | R/W |
| 1004 | Quantity | UINT16 | 0-65535 | R/W |
| 1005 | Color | UINT16 | 0-7 | R/W |
| 1006 | Ring Effect | UINT16 | see Ring Effects | R/W |
| 1007 | Text char 0 | UINT16 | 0-255 (ASCII) | R/W |
| 1008 | Text char 1 | UINT16 | 0-255 (ASCII) | R/W |
| 1009 | Text char 2 | UINT16 | 0-255 (ASCII) | R/W |
| 1010 | Text char 3 | UINT16 | 0-255 (ASCII) | R/W |
| 1011 | Special Command | UINT16 | see Special Commands | R/W |
| 1012 | Button Rotation | UINT16 | 0-1 | R/W |
| 1013 | Assign Address | UINT16 | see below | R/W |
| 1014-1019 | Reserved | - | - | - |
| 1020 | SMC Apply/Trigger | UINT16 | 0/1 | W |
| 1100-1228 | Hub Polling Channels | UINT16 | see below | R/W |

Following the MQTT protocol, `CommandType` (register 1001) and `Offset` (register 1002) are kept as two separate fields. Together they select which payload registers are evaluated on Apply. The hub polling channels occupy a separate block (registers 1100-1228), described further below.

**SMC Address (Register 1000):** Target address of the command.

| Value | Meaning |
|---|---|
| 0 | HUB (SEH itself) |
| 0xFFFF | ALL_SENSORS (broadcast to all buttons) |
| other | Specific button address (device address) |

**SMC CommandType (Register 1001):** Mirrors the `CommandType` field of the MQTT protocol.

| Value | CommandType |
|---|---|
| 0 | - (idle, nothing sent) |
| 1 | SET_PARAMETER |
| 2 | SET_STATUS |

*The SMC protocol additionally defines GET_STATUS, SET_UPDATE, GET_INTERNAL_STATUS and SET_INTERNAL_STATUS. These are not yet mapped and can be added later.*

**SMC Offset (Register 1002):** Mirrors the `Offset` field of the MQTT protocol.

| Value | Offset |
|---|---|
| 0 | (none / empty) |
| 1 | PRE_BUTTON_MODE |
| 2 | POST_BUTTON_MODE |
| 3 | BUTTON_ROTATE |
| 4 | SPECIAL_COMMANDS |
| 5 | ADDRESS_TO_BE_ASSIGNED |

**Valid combinations:** On Apply, `CommandType` and `Offset` together determine which payload registers are evaluated.

| CommandType | Offset | Address | Payload registers |
|---|---|---|---|
| SET_PARAMETER | PRE_BUTTON_MODE | any | Button Mode, Quantity, Color, Ring Effect, Text (1003-1010) |
| SET_PARAMETER | POST_BUTTON_MODE | any | Button Mode, Quantity, Color, Ring Effect, Text (1003-1010) |
| SET_PARAMETER | BUTTON_ROTATE | any | Button Rotation (1012) |
| SET_PARAMETER | (none) | HUB (0) | Hub Polling Channels (1100-1228) |
| SET_STATUS | SPECIAL_COMMANDS | any | Special Command (1011) |
| SET_STATUS | ADDRESS_TO_BE_ASSIGNED | any | Assign Address (1013) |

### Button Mode (Register 1003):

| Value | Meaning |
|---|---|
| 0 | DISABLED |
| 1 | ENABLED |
| 2 | LONGPRESS_ENABLED |

### Color (Register 1005):

| Value | Meaning | Value | Meaning |
|---|---|---|---|
| 0 | COLOFF | 4 | COLYELLOW |
| 1 | COLRED | 5 | COLMAGENTA |
| 2 | COLGREEN | 6 | COLCYAN |
| 3 | COLBLUE | 7 | COLWHITE |

### Ring Effect (Register 1006):

| Value | Meaning | Value | Meaning |
|---|---|---|---|
| 0 | SOLID_RING | 94 | ANIMATED_ARROW_UP |
| 1 | FLASH_RING | 95 | ANIMATED_ARROW_DOWN |
| 2 | CONFIRM | 96 | ANIMATED_ARROW_LEFT |
| 84 | POINT_ANIMATED_CLOCK | 97 | ANIMATED_ARROW_RIGHT |
| 85 | CIRCLE_ANIMATED_CLOCK | 102 | SOLID_ARROW_UP_LEFT |
| 86 | SOLID_ARROW_UP | 103 | SOLID_ARROW_UP_RIGHT |
| 87 | SOLID_ARROW_DOWN | 104 | SOLID_ARROW_DOWN_LEFT |
| 88 | SOLID_ARROW_LEFT | 105 | SOLID_ARROW_DOWN_RIGHT |
| 89 | SOLID_ARROW_RIGHT | 106 | FLASH_ARROW_UP_LEFT |
| 90 | FLASH_ARROW_UP | 107 | FLASH_ARROW_UP_RIGHT |
| 91 | FLASH_ARROW_DOWN | 108 | FLASH_ARROW_DOWN_LEFT |
| 92 | FLASH_ARROW_LEFT | 109 | FLASH_ARROW_DOWN_RIGHT |
| 93 | FLASH_ARROW_RIGHT | | |

**Special Command (Register 1011):** Only effective when CommandType = SET_STATUS and Offset = SPECIAL_COMMANDS.

| Value | Meaning |
|---|---|
| 0 | PRE_PRESSED_STATE |
| 1 | POST_PRESSED_STATE |
| 2 | REBOOT |
| 4 | SHOW_CURRENT_ADDRESS |
| 5 | SELF_TRIGGER |
| 6 | LONG_PRESSED_STATE |

**Text (Register 1007-1010):** Up to 4 ASCII characters for the button display, one character per register (register 1007 = first character). The character is stored in the low byte; the high byte is ignored. Unused characters (value `0`) are internally padded with `@`. Only relevant for ring effects with text display.

**Button Rotation (Register 1012):** Only effective when CommandType = SET_PARAMETER and Offset = BUTTON_ROTATE. Sets the display/button orientation.

| Value | Meaning |
|---|---|
| 0 | BUTTON_ROTATION_NORMAL |
| 1 | BUTTON_ROTATION_UPSIDEDOWN |

**Assign Address (Register 1013):** Only effective when CommandType = SET_STATUS and Offset = ADDRESS_TO_BE_ASSIGNED. Assigns a device address to the addressed button(s).

| Value | Meaning |
|---|---|
| 0 | NEXT (auto-assign next free address) |
| other | Explicit device address to assign |

**Hub Polling Channels (Register 1100-1228):** Only effective when CommandType = SET_PARAMETER, Offset = (none) and Address = HUB (0). Defines which button addresses the HUB (SEH) actively polls. Up to **128** addresses are supported.

| Register | Description | Data Type | Range |
|---|---|---|---|
| 1100 | Polling Channel Count | UINT16 | 0-128 |
| 1101 | Polling Channel 0 | UINT16 | button address |
| 1102 | Polling Channel 1 | UINT16 | button address |
| ... | ... | ... | ... |
| 1228 | Polling Channel 127 | UINT16 | button address |

Polling channel K is located at register `1101 + K` (K = 0..127). Register 1100 holds the number of valid channels. As a single Modbus request is limited to ~123 registers, a long channel list must be written in multiple Function-16 requests before setting the Apply trigger (register 1020).

The maximum number of polling addresses that can actually be used is product-dependent and may be lower than the 128 registers reserved here.

**SMC Apply/Trigger (Register 1020):** After writing the SMC registers, register 1020 must be set to `1` to send the command to the buttons. The register is automatically reset to `0`. The SMC Apply is independent of the LED Apply (register 2).

---

## Input Registers (Function 04) - Status/Read-Only

| Register | Description | Data Type | Range |
|---|---|---|---|
| 0 | Firmware Version Major | UINT16 | 0-255 |
| 1 | Firmware Version Minor | UINT16 | 0-255 |
| 2 | Firmware Version Patch | UINT16 | 0-255 |
| 3 | Number of Configured Strips | UINT16 | 0-5 |
| 4 | Strip 1 - Length (LEDs) | UINT16 | 0-312 |
| 5 | Strip 2 - Length (LEDs) | UINT16 | 0-312 |
| 6 | Strip 3 - Length (LEDs) | UINT16 | 0-312 |
| 7 | Strip 4 - Length (LEDs) | UINT16 | 0-312 |
| 8 | Strip 5 - Length (LEDs) | UINT16 | 0-312 |
| 9 | Power Supply Limit (W×10) | UINT16 | 0-65535 |
| 10 | Total Active LEDs | UINT16 | 0-65535 |
| 11 | Error Code | UINT16 | 0-4 |

**Error Codes:**

| Code | Meaning |
|---|---|
| 0 | No error |
| 1 | No network |
| 2 | No MQTT connection |
| 3 | No configuration |
| 4 | Normal operation |

---

## SMC Button Status (Input Registers)

The states of the connected SMC buttons can be read back via input registers. This corresponds to the button feedback published over MQTT on `/Pub/Data/Smc`. The SMC status range starts at input register address **190**.

Because Modbus has no server-initiated push, status changes must be detected by polling (see also the Notes section). To avoid reading the full status block cyclically, the **SMC Event Counter** is provided.

## SMC Event Counter

| Register | Description | Data Type | Range |
|---|---|---|---|
| 100 | SMC Event Counter | UINT16 | 0-65535 |

The counter is incremented on every SMC button status change and wraps around at 65535. A client can poll this single register cheaply and only re-read the button status blocks when the value has changed. Registers 101-109 are reserved.

## Button Status Registers

The number of mapped buttons equals `MAX_SMC_COUNT`. This value depends on the build target (e.g. 5 for standard variants, up to 128 for certain devices). Each button occupies a block of 4 registers. The base address for button K (K = 0 .. MAX_SMC_COUNT-1) is calculated as:

**Button Base Address = 110 + (K * 4)**

| Offset | Description | Data Type | Range |
|---|---|---|---|
| 0 | Button Address | UINT16 | 0-65535 |
| 1 | Press Count | UINT16 | 0-65535 |
| 2 | Press State | UINT16 | see below |
| 3 | Reserved | - | - |

A Button Address of `0` indicates an unused/empty slot.

**Press State (Offset 2):**

| Value | Meaning |
|---|---|
| 0 | PRE_PRESSED_STATE |
| 1 | POST_PRESSED_STATE |
| 3 | LONGPRESS_STATE |

| Button | Address | Press Count | Press State |
|---|---|---|---|
| 0 | 110 | 111 | 112 |
| 1 | 114 | 115 | 116 |
| 2 | 118 | 119 | 120 |
| ... | ... | ... | ... |
| K | 110 + K×4 | 111 + K×4 | 112 + K×4 |

---

# Available LED Effects

| Effect ID | Name | Description |
|---|---|---|
| 1 | Static | All LEDs lit at constant color |
| 2 | Blink | On/off blinking |
| 3 | Flash | Quick bright pulses |
| 4 | Leading Lines | Running lines forward |
| 5 | Leading Lines Reverse | Running lines backward |
| 6 | Multipicker | Cycling through multiple colors |
| 7 | Pick | Animation from center outward |
| 8 | Place | Animation from outside to center |
| 200 | Knight Rider | Bouncing bar animation |

# Register Mapping - Example

## Activate Strip 1, Segment 0 (green, static)

The following registers must be written (Function 16 - Write Multiple Registers):

| Register | Value | Description |
|---|---|---|
| 100 | 1 | Strip 1 Active = true |
| 101 | 1 | Strip 1 Segment Count = 1 |
| 110 | 0 | Segment 0: StartLED = 0 |
| 111 | 20 | Segment 0: StopLED = 20 |
| 112 | 1 | Segment 0: Effect = Static |
| 113 | 200 | Segment 0: Speed = 200 |
| 114 | 0 | Segment 0: Color1 R = 0 |
| 115 | 150 | Segment 0: Color1 G = 150 |
| 116 | 0 | Segment 0: Color1 B = 0 |
| 117 | 0 | Segment 0: Color2 R = 0 |
| 118 | 0 | Segment 0: Color2 G = 0 |
| 119 | 0 | Segment 0: Color2 B = 0 |
| 120 | 0 | Segment 0: Color3 R = 0 |
| 121 | 0 | Segment 0: Color3 G = 0 |
| 122 | 0 | Segment 0: Color3 B = 0 |
| 2 | 1 | Apply/Trigger |

This corresponds to the following MQTT message:

```
{
  "LED_STRIP_1": {
    "Active": true,
    "Segments": [
      {
        "StartLED": 0,
        "StopLED": 20,
        "Effect": 1,
        "Speed": 200,
        "Colors": [{ "R": 0, "G": 150, "B": 0 }]
      }
    ]
  }
}
```

## SMC Button: Set PRE_BUTTON_MODE (all buttons, green, flashing ring)

The following registers must be written (Function 16 - Write Multiple Registers):

| Register | Value | Description |
|---|---|---|
| 1000 | 0xFFFF | Address = ALL_SENSORS (broadcast) |
| 1001 | 1 | CommandType = SET_PARAMETER |
| 1002 | 1 | Offset = PRE_BUTTON_MODE |
| 1003 | 1 | Button Mode = ENABLED |
| 1004 | 5 | Quantity = 5 |
| 1005 | 2 | Color = COLGREEN |
| 1006 | 1 | Ring Effect = FLASH_RING |
| 1020 | 1 | SMC Apply/Trigger |

This corresponds to the following MQTT message:

```
{
  "Address": "ALL_SENSORS",
  "CommandType": "SET_PARAMETER",
  "Offset": "PRE_BUTTON_MODE",
  "Payload": "ENABLED/5/COLGREEN/FLASH_RING/@@@@"
}
```

## SMC Button: Trigger Special Command (PRE_PRESSED_STATE, all buttons)

| Register | Value | Description |
|---|---|---|
| 1000 | 0xFFFF | Address = ALL_SENSORS (broadcast) |
| 1001 | 2 | CommandType = SET_STATUS |
| 1002 | 4 | Offset = SPECIAL_COMMANDS |
| 1011 | 0 | Special Command = PRE_PRESSED_STATE |
| 1020 | 1 | SMC Apply/Trigger |



This corresponds to the following MQTT message:

```
{
  "Address": "ALL_SENSORS",
  "CommandType": "SET_STATUS",
  "Offset": "SPECIAL_COMMANDS",
  "Payload": "PRE_PRESSED_STATE"
}
```

## SMC Button: Set Button Rotation (all buttons, upside down)

| Register | Value | Description |
|---|---|---|
| 1000 | 0xFFFF | Address = ALL_SENSORS (broadcast) |
| 1001 | 1 | CommandType = SET_PARAMETER |
| 1002 | 3 | Offset = BUTTON_ROTATE |
| 1012 | 1 | Button Rotation = BUTTON_ROTATION_UPSIDEDOWN |
| 1020 | 1 | SMC Apply/Trigger |

This corresponds to the following MQTT message:

```
{
  "Address": "ALL_SENSORS",
  "CommandType": "SET_PARAMETER",
  "Offset": "BUTTON_ROTATE",
  "Payload": "BUTTON_ROTATION_UPSIDEDOWN"
}
```

## SMC Button: Assign Address (all buttons, address 0x1000)

| Register | Value | Description |
|---|---|---|
| 1000 | 0xFFFF | Address = ALL_SENSORS (broadcast) |
| 1001 | 2 | CommandType = SET_STATUS |
| 1002 | 5 | Offset = ADDRESS_TO_BE_ASSIGNED |
| 1013 | 0x1000 | Assign Address = 0x1000 |
| 1020 | 1 | SMC Apply/Trigger |

This corresponds to the following MQTT message:

```
{
  "Address": "ALL_SENSORS",
  "CommandType": "SET_STATUS",
  "Offset": "ADDRESS_TO_BE_ASSIGNED",
  "Payload": "1000"
}
```

> Set register 1013 to `0` to assign the next free address automatically (MQTT payload `"NEXT"`).

## SMC HUB: Configure Polling Channels (poll buttons 0x1000-0x1003)

| Register | Value | Description |
|---|---|---|
| 1000 | 0 | Address = HUB |
| 1001 | 1 | CommandType = SET_PARAMETER |
| 1002 | 0 | Offset = (none) |
| 1100 | 4 | Polling Channel Count = 4 |
| 1101 | 0x1000 | Polling channel 0 |
| 1102 | 0x1001 | Polling channel 1 |
| 1103 | 0x1002 | Polling channel 2 |
| 1104 | 0x1003 | Polling channel 3 |
| 1020 | 1 | SMC Apply/Trigger |

This corresponds to the following MQTT message:

```
{
  "Address": "HUB",
  "CommandType": "SET_PARAMETER",
  "Offset": "",
  "Payload": "1000/1001/1002/1003"
}
```

# Address Calculation - Quick Reference

```
Strip N Base Address:    100 * N          (N = 1..5)
Strip N Active:          100 * N + 0
Strip N Segment Count:   100 * N + 1
Segment M Base Address:  100 * N + 10 + (M * 11)   (M = 0..7)

SMC block (holding):     selector 1000-1002, payload 1003-1013, Apply = 1020
SMC hub polling (holding): count 1100, channel K = 1101 + K   (K = 0..127)
SMC event counter (input): 100
SMC status (input):      button K base = 110 + (K * 4)   (K = 0 ..
MAX_SMC_COUNT-1)
```

| Strip | Active | Seg Count | Seg 0 | Seg 1 | Seg 2 | Seg 3 |
|---|---|---|---|---|---|---|
| 1 | 100 | 101 | 110 | 121 | 132 | 143 |
| 2 | 200 | 201 | 210 | 221 | 232 | 243 |
| 3 | 300 | 301 | 310 | 321 | 332 | 343 |
| 4 | 400 | 401 | 410 | 421 | 432 | 443 |
| 5 | 500 | 501 | 510 | 521 | 532 | 543 |

# Notes

- **Parallel MQTT/Modbus Operation:** Both interfaces write to the same LED strips. The most recently received command (whether MQTT or Modbus) determines the current state.
- **Power Limit:** The power limit applies globally. Segments that would exceed the power limit will not be activated (identical behavior to MQTT).
- **StopLED >= StartLED:** If StopLED < StartLED, StopLED is automatically set to StartLED.
- **Colors:** The sum of R+G+B per color is limited to a maximum of 180 (per-LED current limiting).
- **Unused Colors:** Set unused color registers (Color 2, Color 3) to 0/0/0.
- **Apply Register:** Changes to strip/segment registers only take effect when `1` is written to register 2 (Apply). This allows atomic configuration of multiple strips/segments.
- **SMC Control:** The SMC block (from register 1000) has its own Apply trigger (register 1020), independent of the LED Apply (register 2). LED and SMC commands can thus be triggered separately.
- **SMC Parallel to MQTT:** Modbus and MQTT SMC commands internally generate the same SMC command and are placed into the same send queue. The most recently received command determines the button state.
- **SMC Addressing:** `0xFFFF` addresses all buttons (broadcast), `0` the HUB (SEH itself), any other value a specific button.
- **SMC Status:** The button status input registers (from address 110) reflect the most recently received button feedback. A Button Address of `0` marks an unused slot.
- **Status Change Notification:** Modbus has no server-initiated push; the server only answers requests. Status changes are detected by polling the SMC Event Counter (register

100) and re-reading the button blocks when it changed. For true push notifications use the MQTT channel (`/Pub/Data/Smc`), which runs in parallel.