EMS switchPrograms for EMS+ and and EMS 2.0

Bosch-Group thermostats can manage switchPrograms (SP) for heating circuits, warm water buffer load and circulation pump.

Easy installations with a boiler and continuous-flow water heater and just one heating circuit may use only one SP (e.g. my Moduline2050 with EMS 2.0).
Those thermostats are able to switch between 3 states (levels): “off”, “eco” and “comfort”.
Eco and comfort-temps are maintained separately: ecotemp / comforttemp.

Most installations may use 2 heating circuits, a warm water buffer and a circulation pump.
Using then EMS+ thermostats like RC300/RC310 there are more selectable switchPrograms

For each heating circuit:

Program A (program 1) and Program B (program 2)
and for each program there are 2 switch modes:
- levels (“off”, “eco” and “comfort”) or
- absolute temperatures

For dhw: switchProgram with levels (“off”, “low” and “high”)

For dhw-circ.pump: switchProgram with levels (“off” and “on”)

Each switchProgram has 6 switchTimes per day – for 7 days: 42 switchTimes.
Each switchProgam telegram data has 84 bytes length.

→ Each heating circuit variant for a combination of program and switchprogmode has an own telegram type. (4 telegram types for each hc)

How does Bosch Group (Bosch / Buderus / Junkers / Netfit ...) uses their own gateways?

- The Bosch Group gateways buffer all telegram types available on EMS-bus
- Additionally, the gateway calculates and stores historical data (recordings for temps and energy consumptions) and store them within the gateway:
  - monthly data: actual month and last 12 months
  - daily and hourly values for approx. last 6 weeks

- switchPrograms are maintained and stored within the master thermostat
- Any change from thermostat is send to gateway and buffered there
- The gateway buffers all telegram types even the inactive ones
- The gateway communicates with Bosch cloud and when using one of the Bosch-group apps the app requests data from cloud and then cloud from gateway.
- The gateway answers with the right data selected by switchProgramType and switchProgram by selecting the right telegram type for each heating circuit.

- Any app request for switchPrograms are answered directly from the gateway buffer and no read request to thermostat are send on the ems-bus.
  - just switchProgram changes are send for those days changed from gateway to thermostat when save button is pressed on app.
  - changes for program (A / B) are send to thermostat, but SP is used from gateway buffer
  - changes for switchprogmode are handled in same way – SP used from buffer
- All this minimizes ems-bus traffic to a minimum

- switchPrograms are available in two types: level and absolute temps per switchTime
- the available levels depend on switchProgram type (hc / dhw / cp) and on the actual heating systems installed (e.g. heating pumps levels differ from boiler ones)
- The levels are separate entities returned by BOSCH API for separate for each hc and for dhw. The levels relate to the entities defining the temperatures.

- It is required that the levels do not overlap otherwise the API creates an error message.
  All temps have to be in a range between 5 and 30 °C where
  max or actual value of eco has to be lower than min or actual value of comfort
  This results in min/max values of these entities change when re-defining eco / comfort

- The older gateways allow local LAN API access. (IP-inside, KM50 ...KM300)
  Bosch has stopped to sell these gateways and just promotes the newer ones which do not allow local LAN access anymore (e.g. MX300) and can be just managed by using the apps and cloud services.
- All API messages are encrypted with Rijndael and need to be decrypted / encrypted.
  This is done for local LAN API and as well for communication with Bosch cloud server

What is missing today within EMS-ESP gateway firmware?

- Buffering all telegram types for switchPrograms and holidayModes
- Depending on memory limitations this buffer could be hex or JSON
- Implementing a function to select the "right" switchProgram (telegram type) by using the active parameters for program and switchprogtype for each heating circuit
- The level attributes (hex to text) must be customizable (English language preferred) if not RAW telegrams are used.
- Implementing API / MQTT interface – either for raw hex telegrams or interpreted JSON

- The MQTT climate entities do not contain level attributes. We need to find solution how to manage this, if levels are defined within the EMS-ESP gateway and not within home automation system.
  I would propose separate customizable enum entities for levels per hc and dhw unless decoding is done in home automation system.

How are EMS-ESP switchPrograms implemented within ioBroker today?

The actual version of the ioBroker adapter uses the following logic:

- The switchProgammode (spm) are read by using raw telegram read with offset (the new entiies are not used yet)
- The combination of spm and switchprogram defines the telegram type to be read
- On first init run all 4 possible heatings circuits are tested if telegram type responds. Those entries are enabled, all other disabled for further runs.
- Depending on number SP'S found the adapter polls between every minute or up to every 3 minutes
- On every poll the 84 bytes of each telegram are requested (this takes between 1 and 40 seconds) – Timing is critical
- The telegrams are then encoded and stored as JSON in ioBroker objects/states.
- The adapter can read and write back to thermostat
- The adapter run using the option for discovering / reading SP's creates quite som load on EMS-bus and in some case produces crashes for EMS-ESP gateway

My conclusion

Any regular polling to ems-bus should be avoided to secure stability with low bus error rate.

All existing telegram types should be stored within EMS-ESP gateway.
This could be raw telegram data (hex) (1st step) and/or decoded data (JSON) (2nd step)

The decoding from hex telegram to JSON data is critical for levels.
The translation from hex to level-text must be consistent to existing installation and should be customizable for each telegram type if encoding should be done within EMS-ESP.

My prototype implementation of switchPrograms within Home Assistant (HA)

I implemented a prototype for managing EMS-bus switchPrograms within HA using the HASS integration for scheduler component and scheduler custom card. (read / write)

I implemented this for my EMS2.0 installation with just one SP and as well for my EMS+ installation with RC310.

The prototype allows reading the telegrams from EMS-ESP by using raw telegram send/request by MQTT and translating the received telegrams to scheduler data. Changed data could be written back to EMS-ESP using raw hex telegrams.

The following logic is implemented using Node-Red flows:

- Identifying the right telegram type by reading program and switchprogmode
- Defining one timeslot (daily) or two workdays / weekend
- Reading one or two telegrams with 12 bytes (one day) for the selected telegram type Monday (offset “0x00”) to represent working week or to represent the complete week. Saturday (offset “0x3C”) to represent weekend
- The telegrams read requests are initiated by a HA button to be pressed when scheduler entities should be added (1st time) or updated with thermostat data.
- The telegram data received per day is then decoded and translated to scheduler entity data structure required:
  - weekly schedule: daily / workdays / or weekends
  - time slots
  - services to be called in HA
  - services data: temperatures or levels
- The scheduler component requires all entities to be added or edited by own service calls. The entities cannot be added by MQTT!
- Each scheduler entity is a switch with additional data and can be switched on or off. Each time variant needs an own entity. For different schedules per day 7 entities would be needed and need to be created by Node-red flows.
  I do not need this – 2 differentiating between workdays (mon-fri) and weekend (sat-sun) are sufficient.
- Initiated by another push button the scheduler data could be read and translated back into a raw telegram to be send to EMS-ESP by using MQTT.

Critical points for this flow and using the scheduler component are:

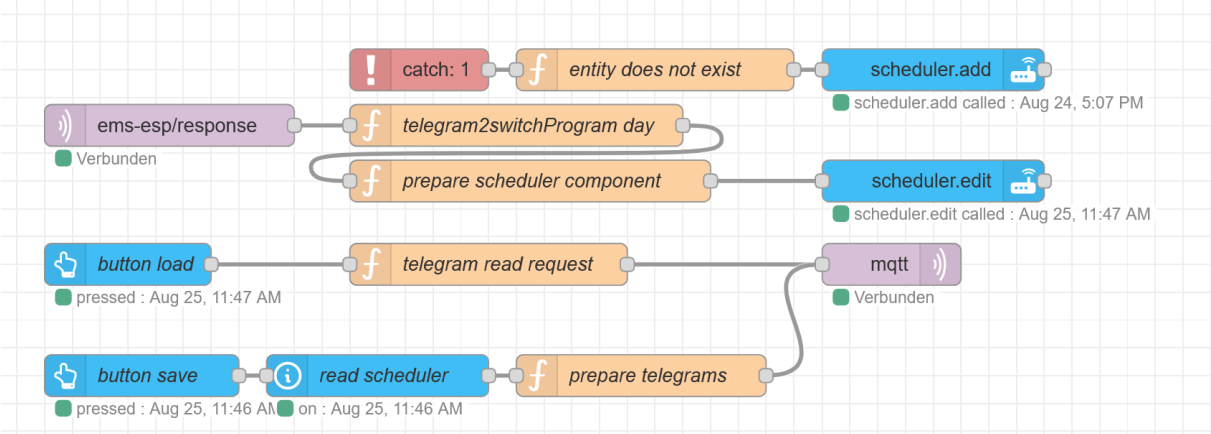
- Selecting the right entity and service call for scheduler component for each telegram type

- For heating circuits and switchprogmode with absolute temps it could be:
  climate.thermostat_hcx with service climate.set_temperature
- For heating circuits and switchprogmode with levels the climate entity does not work, since the levels are not defined as hvac options.
  There is a need to define an own input_select.xxxx entity with the right options.
  Service call then input_select.select_option
- When using scheduler for dhw the existing entity select.thermostat_dhw_mode can be used with service call select.select_option. In my case the existing options do not match. “eco” is here “Normal”. (see customizing texts)
- The existing select options are language dependend – this is not ideal.
  Maybe better to define own input_select items in HA and use them or implement customization in EMS-ESP.

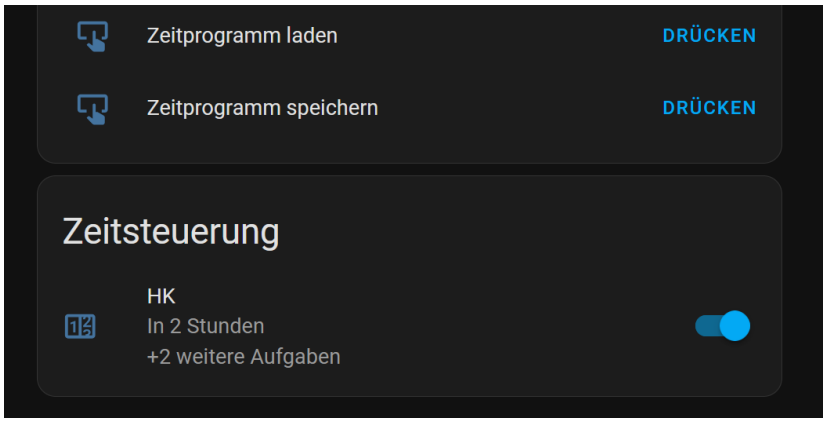
- When using one of the original climate entities the schedule entities will switch to manual when switched on, but not back to automatic when switched off.
  This has to be managed by own NR flow.

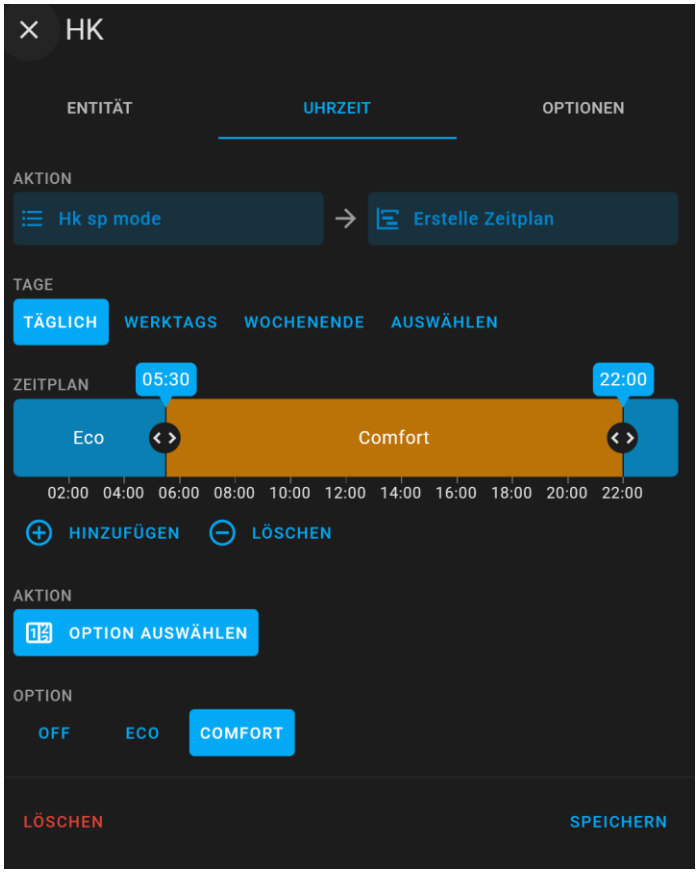
- The scheduler entities have to be added / edited by using the HA scheduler service calls. Therefore these entities can’t be added by using MQTT discovery function.
- Each time variant per day needs an own entity.
  I haven’t found a Lovelace card to show the weekly switchTimes in one overview.
  (the Bosch app is better in this respect).

The simplest flow for my holiday house with just one switchProgram looks like:



And in Lovelace:





Appendix: Telegram types and structure:

RC310 thermostat:

Each heating circuit has 2 switchPrograms A and B and 2 variants with level and temps.

The heating circuits can be set to use levels (off/eco/comfort) or absolute temps per switchProgram.

The switch is done by the entity hcx/switchprogmode for each heating circuit between levels and absolute (temps). These entities have been build-in just recently.

Telegram types: (in RAW hex – as used in NR flows)

Program A (1):
hc1: level: 0x01C3 / absolute: 0x0583
hc2: level: 0x01C4 / absolute: 0x0584

Program B (2):
hc1: level: 0x0349 / absolute: 0x058D
hc2: level: 0x034A / absolute: 0x058E

DHW:

0x01FF for warm water buffer load
0x0209 for circulation pump

Telegram structure:

per day there is a maximum of 6 switch points. Each switchpoint is represented by 2 bytes. Therefore, each day is 12 bytes long:

Mo offset	0	0x00
Tu offset	12	0x0C
We offset	24	0x18
Th offset	36	0x24
Fr offset	48	0x30
Sa offset	60	0x3C
Su offset	72	0x48

Telegram structure of one day looks like this with 2 active switchpoints:

P1 P2 P3 P4 P5 P6 .....P12

14 01 58 03 FF 03 FF 03 FF 03 FF 03

P1: time of 1st sp: unit 15 minutes --> 0x14 = 20 *15 = 300 minutes = 05:00 hours
FF indicates that switchTime is not active
P2: temperature or level : temps: Number("0x" + hex) / 2 range (5 ... 30)

hc levels: 00 : off, 01 : eco, 03 : comfort / 02 = eco+ for heatpumps ???
ww levels: 00 : off, 01 : low, 02 : high
cp levels: 00: off, FF : on