

```
1  /**
2  *
3  *  @copyright &copy; 2010 - 2020, Fraunhofer-Gesellschaft zur Foerderung der
4  *  angewandten Forschung e.V. All rights reserved.
5  *
6  *  BSD 3-Clause License
7  *  Redistribution and use in source and binary forms, with or without
8  *  modification, are permitted provided that the following conditions are met:
9  *  1. Redistributions of source code must retain the above copyright notice,
10 *  this list of conditions and the following disclaimer.
11 *  2. Redistributions in binary form must reproduce the above copyright
12 *  notice, this list of conditions and the following disclaimer in the
13 *  documentation and/or other materials provided with the distribution.
14 *  3. Neither the name of the copyright holder nor the names of its
15 *  contributors may be used to endorse or promote products derived from
16 *  this software without specific prior written permission.
17 *
18 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 *  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 *  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 *  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 *  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 *  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 *  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 *  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 *  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 *  POSSIBILITY OF SUCH DAMAGE.
29 *
30 *  We kindly request you to use one or more of the following phrases to refer
31 *  to foxBMS in your hardware, software, documentation or advertising
32 *  materials:
33 *
34 *  &Prime;This product uses parts of foxBMS&reg;&Prime;;
35 *
36 *  &Prime;This product includes parts of foxBMS&reg;&Prime;;
37 *
38 *  &Prime;This product is derived from foxBMS&reg;&Prime;;
39 *
40 */
41
42 /**
43 *  @file    database_cfg.h
44 *  @author  foxBMS Team
45 *  @date    18.08.2015 (date of creation)
46 *  @ingroup ENGINE_CONF
47 *  @prefix  DATA
48 *
49 *  @brief   Database configuration header
50 *
51 *  Provides interfaces to database configuration
52 *
```

```

53     */
54
55 #ifndef DATABASE_CFG_H_
56 #define DATABASE_CFG_H_
57
58 /*===== Includes =====*/
59 #include "general.h"
60
61 #include "batterysystem_cfg.h"
62
63 /*===== Macros and Definitions =====*/
64
65 /**
66  * @brief maximum amount of data block
67  *
68  * this value is extendible but limitation is done due to RAM consumption and performance
69  */
70 #define DATA_MAX_BLOCK_NR          25          /* max 25 Blocks currently supported*/
71
72 /**
73  * @brief data block identification number
74  */
75 typedef enum {
76     DATA_BLOCK_00          = 0,
77     DATA_BLOCK_01          = 1,
78     DATA_BLOCK_02          = 2,
79     DATA_BLOCK_03          = 3,
80     DATA_BLOCK_04          = 4,
81     DATA_BLOCK_05          = 5,
82     DATA_BLOCK_06          = 6,
83     DATA_BLOCK_07          = 7,
84     DATA_BLOCK_08          = 8,
85     DATA_BLOCK_09          = 9,
86     DATA_BLOCK_10          = 10,
87     DATA_BLOCK_11          = 11,
88     DATA_BLOCK_12          = 12,
89     DATA_BLOCK_13          = 13,
90     DATA_BLOCK_14          = 14,
91     DATA_BLOCK_15          = 15,
92     DATA_BLOCK_16          = 16,
93     DATA_BLOCK_17          = 17,
94     DATA_BLOCK_18          = 18,
95     DATA_BLOCK_19          = 19,
96     DATA_BLOCK_20          = 20,
97     DATA_BLOCK_21          = 21,
98     DATA_BLOCK_22          = 22,
99     DATA_BLOCK_23          = 23,
100    DATA_BLOCK_24          = 24,
101    DATA_BLOCK_MAX         = DATA_MAX_BLOCK_NR,
102 } DATA_BLOCK_ID_TYPE_e;
103
104

```

```

105  /**
106   * @brief data block access types
107   *
108   * read or write access types
109   */
110  typedef enum {
111      WRITE_ACCESS = 0,  /*!< write access to data block */
112      READ_ACCESS  = 1,  /*!< read access to data block */
113  } DATA_BLOCK_ACCESS_TYPE_e;
114
115  /**
116   * configuration struct of database channel (data block)
117   */
118  typedef struct {
119      void *blockptr;
120      uint16_t datalength;
121  } DATA_BASE_HEADER_s;
122
123  /**
124   * configuration struct of database device
125   */
126  typedef struct {
127      uint8_t nr_of_blockheader;
128      DATA_BASE_HEADER_s *blockheaderptr;
129  } DATA_BASE_HEADER_DEV_s;
130
131
132  /**
133   * Definitions for each database entry
134   */
135  #define DATA_BLOCK_ID_CELLVOLTAGE          DATA_BLOCK_00
136  #define DATA_BLOCK_ID_CELLTEMPERATURE     DATA_BLOCK_01
137  #define DATA_BLOCK_ID_SOX                  DATA_BLOCK_02
138  #define DATA_BLOCK_ID_BALANCING_CONTROL_VALUES DATA_BLOCK_03
139  #define DATA_BLOCK_ID_BALANCING_FEEDBACK_VALUES DATA_BLOCK_04
140  #define DATA_BLOCK_ID_CURRENT_SENSOR      DATA_BLOCK_05
141  #define DATA_BLOCK_ID_HW_INFO              DATA_BLOCK_06
142  #define DATA_BLOCK_ID_STATEREQUEST         DATA_BLOCK_07
143  #define DATA_BLOCK_ID_MINMAX               DATA_BLOCK_08
144  #define DATA_BLOCK_ID_ISOGUARD             DATA_BLOCK_09
145  #define DATA_BLOCK_ID_SLAVE_CONTROL       DATA_BLOCK_10
146  #define DATA_BLOCK_ID_OPEN_WIRE           DATA_BLOCK_11
147  #define DATA_BLOCK_ID_LTC_DEVICE_PARAMETER DATA_BLOCK_12
148  #define DATA_BLOCK_ID_LTC_ACCURACY         DATA_BLOCK_13
149  #define DATA_BLOCK_ID_ERRORSTATE           DATA_BLOCK_14
150  #define DATA_BLOCK_ID_MSL                  DATA_BLOCK_15
151  #define DATA_BLOCK_ID_RSL                  DATA_BLOCK_16
152  #define DATA_BLOCK_ID_MOL                  DATA_BLOCK_17
153  #define DATA_BLOCK_ID_MOV_AVERAGE         DATA_BLOCK_18
154  #define DATA_BLOCK_ID_CONTFEEDBACK         DATA_BLOCK_19
155  #define DATA_BLOCK_ID_ILCKFEEDBACK         DATA_BLOCK_20
156  #define DATA_BLOCK_ID_SYSTEMSTATE          DATA_BLOCK_21

```

```

157 #define DATA_BLOCK_ID_SOF DATA_BLOCK_22
158 #define DATA_BLOCK_ID_ALLGPIOVOLTAGE DATA_BLOCK_23
159 #define DATA_BLOCK_ID_CONT_SOH DATA_BLOCK_24
160
161 /**
162  * data block struct of cell voltage
163  */
164 typedef struct {
165     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
166     uint32_t timestamp; /*!< timestamp of database entry */
167     uint32_t previous_timestamp; /*!< timestamp of last database entry */
168     uint16_t voltage[BS_NR_OF_BAT_CELLS]; /*!< unit: mV */
169     uint32_t valid_volt[BS_NR_OF_MODULES]; /*!< bitmask if voltages are valid. 0->valid, 1->invalid */
170     uint32_t sumOfCells[BS_NR_OF_MODULES]; /*!< unit: mV */
171     uint8_t valid_socPECs[BS_NR_OF_MODULES]; /*!< 0 -> if PEC okay; 1 -> PEC error */
172     uint32_t packVoltage_mV; /*!< uint: mV */
173     uint8_t state; /*!< for future use */
174 } DATA_BLOCK_CELLVOLTAGE_s;
175
176 /**
177  * data block struct of cell voltage
178  */
179 typedef struct {
180     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
181     uint32_t timestamp; /*!< timestamp of database entry */
182     uint32_t previous_timestamp; /*!< timestamp of last database entry */
183     uint8_t openwire[BS_NR_OF_MODULES * (BS_NR_OF_BAT_CELLS_PER_MODULE+1)]; /*!< 1 -> open wire, 0 -> everything ok */
184     uint8_t state; /*!< for future use */
185 } DATA_BLOCK_OPENWIRE_s;
186
187 /**
188  * data block struct of cell temperatures
189  */
190 typedef struct {
191     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
192     uint32_t timestamp; /*!< timestamp of database entry */
193     uint32_t previous_timestamp; /*!< timestamp of last database entry */
194     int16_t temperature[BS_NR_OF_TEMP_SENSORS]; /*!< unit: degree Celsius */
195     uint16_t valid_temperature[BS_NR_OF_MODULES]; /*!< bitmask if temperatures are valid. 0->valid, 1->invalid */
196     uint8_t state; /*!< for future use */
197 } DATA_BLOCK_CELLTEMPERATURE_s;
198
199 /**
200  * data block struct of sox
201  */
202 typedef struct {
203     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
204     uint32_t timestamp; /*!< timestamp of database entry */
205     uint32_t previous_timestamp; /*!< timestamp of last database entry */
206     float soc_mean; /*!< 0.0 <= soc_mean <= 100.0 */
207     float soc_min; /*!< 0.0 <= soc_min <= 100.0 */

```

```

208     float soc_max;                                /*!< 0.0 <= soc_max <= 100.0 */
209     uint8_t state;                                /*!< */
210 } DATA_BLOCK_SOX_s;
211
212
213 /**
214  * data block struct of sof limits
215  */
216 typedef struct {
217     uint32_t timestamp;                            /*!< timestamp of database entry */
218     uint32_t previous_timestamp;                    /*!< timestamp of last database entry */
219     float recommended_continuous_charge;            /*!< recommended continuous operating charge current */
220     float recommended_continuous_discharge;         /*!< recommended continuous operating discharge current */
221     float recommended_peak_charge;                 /*!< recommended peak operating charge current */
222     float recommended_peak_discharge;              /*!< recommended peak operating discharge current */
223     float continuous_charge_MOL;                   /*!< charge current maximum operating level */
224     float continuous_discharge_MOL;                /*!< discharge current maximum operating level */
225     float continuous_charge_RSL;                   /*!< charge current recommended safety level */
226     float continuous_discharge_RSL;                /*!< discharge current recommended safety level */
227     float continuous_charge_MSL;                   /*!< charge current maximum safety level */
228     float continuous_discharge_MSL;                /*!< discharge current maximum safety level */
229 } DATA_BLOCK_SOF_s;
230
231
232 /**
233  * data structure declaration of DATA_BLOCK_BALANCING_CONTROL
234  */
235 typedef struct {
236     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
237     uint32_t timestamp;                            /*!< timestamp of database entry */
238     uint32_t previous_timestamp;                    /*!< timestamp of last database entry */
239     uint8_t balancing_state[BS_NR_OF_BAT_CELLS];    /*!< 0 means balancing is active, 0 means balancing is inactive*/
240     uint32_t delta_charge[BS_NR_OF_BAT_CELLS];     /*!< Difference in Depth-of-Discharge in mAs*/
241     uint8_t enable_balancing;                       /*!< Switch for enabling/disabling balancing */
242     uint8_t threshold;                             /*!< balancing threshold in mV */
243     uint8_t request;                               /*!< balancing request per CAN */
244     uint8_t state;                                 /*!< for future use */
245 } DATA_BLOCK_BALANCING_CONTROL_s;
246
247 /**
248  * data structure declaration of DATA_BLOCK_USER_IO_CONTROL
249  */
250 typedef struct {
251     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
252     uint32_t timestamp;                            /*!< timestamp of database entry */
253     uint32_t previous_timestamp;                    /*!< timestamp of last database entry */
254     uint8_t io_value_out[BS_NR_OF_MODULES];         /*!< data to be written to the port expander */
255     uint8_t io_value_in[BS_NR_OF_MODULES];          /*!< data read from to the port expander */
256     uint8_t eeprom_value_write[BS_NR_OF_MODULES];   /*!< data to be written to the slave EEPROM */
257     uint8_t eeprom_value_read[BS_NR_OF_MODULES];    /*!< data read from to the slave EEPROM */
258     uint8_t external_sensor_temperature[BS_NR_OF_MODULES]; /*!< temperature from the external sensor on slave */
259     uint32_t eeprom_read_address_to_use;            /*!< address to read from for slave EEPROM */

```

```

260     uint32_t eeprom_read_address_last_used;                /*!< last address used to read fromfor slave
    EEPROM        */
261     uint32_t eeprom_write_address_to_use;                /*!< address to write to for slave EEPROM        */
262     uint32_t eeprom_write_address_last_used;            /*!< last address used to write to for slave
    EEPROM        */
263     uint8_t state;                                        /*!< for future use        */
264 } DATA_BLOCK_SLAVE_CONTROL_s;
265
266 /**
267  * data block struct of cell balancing feedback
268  */
269 typedef struct {
270     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
271     uint32_t timestamp;                                    /*!< timestamp of database entry        */
272     uint32_t previous_timestamp;                          /*!< timestamp of last database entry    */
273     uint16_t value[BS_NR_OF_MODULES];                    /*!< unit: mV (opto-coupler output)    */
274     uint8_t state;                                        /*!< for future use        */
275 } DATA_BLOCK_BALANCING_FEEDBACK_s;
276
277
278 /**
279  * data block struct of user multiplexer values
280  */
281
282 typedef struct {
283     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
284     uint32_t timestamp;                                    /*!< timestamp of database entry        */
285     uint32_t previous_timestamp;                          /*!< timestamp of last database entry    */
286     uint16_t value[BS_N_MUX_CHANNELS_PER_MUX*BS_N_USER_MUX_PER_LTC*BS_NR_OF_MODULES]; /*!< unit: mV (mux
    voltage input)        */
287     uint8_t state;                                        /*!< for future use        */
288 } DATA_BLOCK_USER_MUX_s;
289
290 /**
291  * data block struct of current measurement
292  */
293 typedef struct {
294     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
295     uint32_t timestamp;                                    /*!< timestamp of database entry        */
296     uint32_t previous_timestamp;                          /*!< timestamp of last database entry    */
297     int32_t current;                                       /*!< unit: mA        */
298     float voltage[BS_NR_OF_VOLTAGES_FROM_CURRENT_SENSOR]; /*!< unit: mV        */
299     float temperature;                                     /*!< unit: 0.1deg;C        */
300     float power;                                           /*!< unit: W        */
301     float current_counter;                                 /*!< unit: A.s        */
302     float energy_counter;                                  /*!< unit: W.h        */
303     uint8_t state_current;
304     uint8_t state_voltage;
305     uint8_t state_temperature;
306     uint8_t state_power;
307     uint8_t state_cc;
308     uint8_t state_ec;

```

```

309     uint8_t newCurrent;
310     uint8_t newPower;
311     uint32_t previous_timestamp_cur;           /*!< timestamp of current database entry */
312     uint32_t timestamp_cur;                   /*!< timestamp of current database entry */
313     uint32_t previous_timestamp_cc;           /*!< timestamp of C-C database entry */
314     uint32_t timestamp_cc;                   /*!< timestamp of C-C database entry */
315 } DATA_BLOCK_CURRENT_SENSOR_s;
316
317 /**
318  * data block struct of hardware info
319  */
320 typedef struct {
321     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
322     uint32_t timestamp;                       /*!< timestamp of database entry */
323     uint32_t previous_timestamp;              /*!< timestamp of last database entry */
324     float vbat_mV;                           /*!< unit: mV */
325     float temperature;                       /*!< unit: degree Celsius */
326     uint8_t state_vbat;                      /*!< */
327     uint8_t state_temperature;               /*!< */
328 } DATA_BLOCK_HW_INFO_s;
329
330 /**
331  * data block struct of can state request
332  */
333
334 typedef struct {
335     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
336     uint32_t timestamp;                       /*!< timestamp of database entry */
337     uint32_t previous_timestamp;              /*!< timestamp of last database entry */
338     uint8_t state_request;
339     uint8_t previous_state_request;
340     uint8_t state_request_pending;
341     uint8_t state;
342 } DATA_BLOCK_STATEREQUEST_s;
343
344 /**
345  * data block struct of LTC minimum and maximum values
346  */
347 typedef struct {
348     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
349     uint32_t timestamp;                       /*!< timestamp of database entry */
350     uint32_t previous_timestamp;              /*!< timestamp of last database entry */
351     uint32_t voltage_mean;
352     uint16_t voltage_min;
353     uint16_t voltage_module_number_min;
354     uint16_t voltage_cell_number_min;
355     uint16_t previous_voltage_min;
356     uint16_t voltage_max;
357     uint16_t voltage_module_number_max;
358     uint16_t voltage_cell_number_max;
359     uint16_t previous_voltage_max;
360     float temperature_mean;

```

```

361     int16_t temperature_min;
362     uint16_t temperature_module_number_min;
363     uint16_t temperature_sensor_number_min;
364     int16_t temperature_max;
365     uint16_t temperature_module_number_max;
366     uint16_t temperature_sensor_number_max;
367     uint8_t state;
368 } DATA_BLOCK_MINMAX_s;
369
370 /**
371  * data block struct of isometer measurement
372  */
373 typedef struct {
374     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
375     uint32_t timestamp; /*!< timestamp of database entry */
376     uint32_t previous_timestamp; /*!< timestamp of last database entry */
377     uint8_t valid; /*!< 0 -> valid, 1 -> resistance unreliable */
378     uint8_t state; /*!< 0 -> resistance/measurement OK , 1 -> resistance too low or error */
379     uint32_t resistance_kOhm; /*!< insulation resistance measured in kOhm */
380 } DATA_BLOCK_ISOMETER_s;
381
382 /**
383  * data block struct of ltc device parameter
384  */
385 typedef struct {
386     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
387     uint32_t timestamp; /*!< timestamp of database entry */
388     uint32_t previous_timestamp; /*!< timestamp of last database entry */
389     uint32_t sumOfCells[BS_NR_OF_MODULES];
390     uint8_t valid_sumOfCells[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 ->
391     unreliable */
392     uint16_t dieTemperature[BS_NR_OF_MODULES]; /* die temperature in degree
393     celsius */
394     uint8_t valid_dieTemperature[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 ->
395     unreliable */
396     uint32_t analogSupplyVolt[BS_NR_OF_MODULES]; /* voltage in
397     [uV] */
398     uint8_t valid_analogSupplyVolt[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 ->
399     unreliable */
400     uint32_t digitalSupplyVolt[BS_NR_OF_MODULES]; /* voltage in
401     [uV] */
402     uint8_t valid_digitalSupplyVolt[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 ->
403     unreliable */
404     uint32_t valid_cellvoltages[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 -> invalid, bit0 -> cell 0, bit1 -> cell 1
405     ... */
406     uint8_t valid_GPIOs[BS_NR_OF_MODULES]; /*!< 0 -> valid, 1 -> invalid, bit0 -> GPIO0, bit1 -> GPIO1
407     ... */
408     uint8_t valid_LTC[BS_NR_OF_MODULES]; /*!< 0 -> LTC working, 1 -> LTC
409     defect */
410 } DATA_BLOCK_LTC_DEVICE_PARAMETER_s;
411
412 /**

```



```

403     * data block struct of ltc adc accuracy measurement
404     */
405     typedef struct {
406         /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
407         uint32_t timestamp;                /*!< timestamp of database entry          */
408         uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
409         int adc1_deviation[BS_NR_OF_MODULES]; /* ADC1 deviation from 2nd reference */
410         int adc2_deviation[BS_NR_OF_MODULES]; /* ADC2 deviation from 2nd reference */
411     } DATA_BLOCK_LTC_ADC_ACCURACY_s;
412
413     /**
414     * data block struct of error flags
415     */
416     typedef struct {
417         /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
418         uint32_t timestamp;                /*!< timestamp of database entry          */
419         uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
420         uint8_t currentsensorresponding;    /*!< 0 -> no error, 1 -> error            */
421         uint8_t main_plus;                 /*!< 0 -> no error, 1 -> error            */
422         uint8_t main_minus;                /*!< 0 -> no error, 1 -> error            */
423         uint8_t precharge;                 /*!< 0 -> no error, 1 -> error            */
424         uint8_t charge_main_plus;          /*!< 0 -> no error, 1 -> error            */
425         uint8_t charge_main_minus;         /*!< 0 -> no error, 1 -> error            */
426         uint8_t charge_precharge;          /*!< 0 -> no error, 1 -> error            */
427         uint8_t interlock;                 /*!< 0 -> no error, 1 -> error            */
428         uint8_t crc_error;                 /*!< 0 -> no error, 1 -> error            */
429         uint8_t mux_error;                 /*!< 0 -> no error, 1 -> error            */
430         uint8_t spi_error;                 /*!< 0 -> no error, 1 -> error            */
431         uint8_t ltc_config_error;           /*!< 0 -> no error, 1 -> error            */
432         uint8_t insulation_error;           /*!< 0 -> no error, 1 -> error            */
433         uint8_t fuse_state_normal;          /*!< 0 -> fuse ok, 1 -> fuse tripped      */
434         uint8_t fuse_state_charge;          /*!< 0 -> fuse ok, 1 -> fuse tripped      */
435         uint8_t open_wire;                 /*!< 0 -> no error, 1 -> error            */
436         uint8_t can_timing;                /*!< 0 -> no error, 1 -> error            */
437         uint8_t can_timing_cc;             /*!< 0 -> no error, 1 -> error            */
438         uint8_t mcuDieTemperature;         /*!< 0 -> no error, 1 -> error            */
439         uint8_t coinCellVoltage;           /*!< 0 -> no error, 1 -> error            */
440         uint8_t plausibilityCheck;          /*!< 0 -> no error, else: error            */
441         uint8_t deepDischargeDetected;      /*!< 0 -> no error, 1 -> error            */
442         uint8_t currentOnOpenPowerline;     /*!< 0 -> no error, 1 -> error            */
443     } DATA_BLOCK_ERRORSTATE_s;
444
445     typedef struct {
446         /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
447         uint32_t timestamp;                /*!< timestamp of database entry          */
448         uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
449         uint8_t over_voltage;              /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
450         uint8_t under_voltage;             /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
451         uint8_t over_temperature_charge;    /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
452         uint8_t over_temperature_discharge; /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
453         uint8_t under_temperature_charge;   /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
454         uint8_t under_temperature_discharge; /*!< 0 -> MSL NOT violated, 1 -> MSL violated */

```

```

455     uint8_t over_current_charge_cell;          /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
456     uint8_t over_current_charge_pl0;          /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
457     uint8_t over_current_charge_pl1;          /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
458     uint8_t over_current_discharge_cell;      /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
459     uint8_t over_current_discharge_pl0;       /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
460     uint8_t over_current_discharge_pl1;       /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
461     uint8_t pcb_over_temperature;             /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
462     uint8_t pcb_under_temperature;            /*!< 0 -> MSL NOT violated, 1 -> MSL violated */
463 } DATA_BLOCK_MSL_FLAG_s;
464
465 typedef struct {
466     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
467     uint32_t timestamp;                       /*!< timestamp of database entry */
468     uint32_t previous_timestamp;              /*!< timestamp of last database entry */
469     uint8_t over_voltage;                     /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
470     uint8_t under_voltage;                    /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
471     uint8_t over_temperature_charge;          /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
472     uint8_t over_temperature_discharge;       /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
473     uint8_t under_temperature_charge;         /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
474     uint8_t under_temperature_discharge;      /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
475     uint8_t over_current_charge_cell;         /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
476     uint8_t over_current_charge_pl0;          /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
477     uint8_t over_current_charge_pl1;          /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
478     uint8_t over_current_discharge_cell;      /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
479     uint8_t over_current_discharge_pl0;       /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
480     uint8_t over_current_discharge_pl1;       /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
481     uint8_t pcb_over_temperature;             /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
482     uint8_t pcb_under_temperature;            /*!< 0 -> RSL NOT violated, 1 -> RSL violated */
483 } DATA_BLOCK_RSL_FLAG_s;
484
485 typedef struct {
486     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
487     uint32_t timestamp;                       /*!< timestamp of database entry */
488     uint32_t previous_timestamp;              /*!< timestamp of last database entry */
489     uint8_t over_voltage;                     /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
490     uint8_t under_voltage;                    /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
491     uint8_t over_temperature_charge;          /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
492     uint8_t over_temperature_discharge;       /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
493     uint8_t under_temperature_charge;         /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
494     uint8_t under_temperature_discharge;      /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
495     uint8_t over_current_charge_cell;         /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
496     uint8_t over_current_charge_pl0;          /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
497     uint8_t over_current_charge_pl1;          /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
498     uint8_t over_current_discharge_cell;      /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
499     uint8_t over_current_discharge_pl0;       /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
500     uint8_t over_current_discharge_pl1;       /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
501     uint8_t pcb_over_temperature;             /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
502     uint8_t pcb_under_temperature;            /*!< 0 -> MOL NOT violated, 1 -> MOL violated */
503 } DATA_BLOCK_MOL_FLAG_s;
504
505 typedef struct {
506     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */

```

```

507     uint32_t timestamp;                /*!< timestamp of database entry          */
508     uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
509     float movAverage_current_1s;        /*!< current moving average over the last 1s      */
510     float movAverage_current_5s;        /*!< current moving average over the last 5s      */
511     float movAverage_current_10s;       /*!< current moving average over the last 10s     */
512     float movAverage_current_30s;       /*!< current moving average over the last 30s     */
513     float movAverage_current_60s;       /*!< current moving average over the last 60s     */
514     float movAverage_current_config;    /*!< current moving average over the last configured time */
515     float movAverage_power_1s;          /*!< power moving average over the last 1s      */
516     float movAverage_power_5s;          /*!< power moving average over the last 5s      */
517     float movAverage_power_10s;         /*!< power moving average over the last 10s     */
518     float movAverage_power_30s;         /*!< power moving average over the last 30s     */
519     float movAverage_power_60s;         /*!< power moving average over the last 60s     */
520     float movAverage_power_config;      /*!< power moving average over the last configured time */
521 } DATA_BLOCK_MOVING_AVERAGE_s;
522
523 /**
524  * data block struct of contactor feedback
525  */
526 typedef struct {
527     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
528     uint32_t timestamp;                /*!< timestamp of database entry          */
529     uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
530     uint16_t contactor_feedback;       /*!< feedback of contactors, without interlock */
531 } DATA_BLOCK_CONTFEEDBACK_s;
532
533 /**
534  * data block struct of interlock feedback
535  */
536 typedef struct {
537     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
538     uint32_t timestamp;                /*!< timestamp of database entry          */
539     uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
540     uint8_t interlock_feedback;        /*!< feedback of interlock, without contactors */
541 } DATA_BLOCK_ILCKFEEDBACK_s;
542
543 /**
544  * data block struct of system state
545  */
546 typedef struct {
547     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
548     uint32_t timestamp;                /*!< timestamp of database entry          */
549     uint32_t previous_timestamp;        /*!< timestamp of last database entry      */
550     uint8_t bms_state;                 /*!< system state (e.g., standby, normal) */
551 } DATA_BLOCK_SYSTEMSTATE_s;
552
553 /**
554  * data block struct of cell voltage
555  */
556 typedef struct {
557     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
558     uint32_t timestamp;                /*!< timestamp of database

```

```

559     entry                                     */
uint32_t previous_timestamp;                    /*!< timestamp of last database
560     entry                                     */
uint16_t gpiovoltage[BS_NR_OF_MODULES * BS_NR_OF_GPIOS_PER_MODULE]; /*!< unit:
mV                                              */
561     uint16_t valid_gpiovoltages[BS_NR_OF_MODULES]; /*!< bitmask if voltages are valid.
0->valid, 1->invalid */
562     uint8_t state; /*!< for future
use                                           */
563 } DATA_BLOCK_ALLGPIOVOLTAGE_s;
564
565 /**
566  * data block struct of contactor SOH
567  */
568 typedef struct {
569     /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
570     uint32_t timestamp; /*!< timestamp of database entry */
571     uint32_t previous_timestamp; /*!< timestamp of last database entry */
572     float contactor_soh[BS_NR_OF_CONTACTORS]; /*!< SOH of contactors */
573 } DATA_BLOCK_CONT_SOH_s;
574
575 /*===== Extern Constant and Variable Declarations =====*/
576
577 /**
578  * @brief device configuration of database
579  *
580  * all attributes of device configuration are listed here (pointer to channel list, number of channels)
581  */
582 extern const DATA_BASE_HEADER_DEV_s data_base_dev;
583
584 /*===== Extern Function Prototypes =====*/
585
586 #endif /* DATABASE_CFG_H_ */
587

```