```
/**
 * @file    database_cfg.h
 * @author  foxBMS Team
 * @date    18.08.2015 (date of creation)
 * @ingroup ENGINE_CONF
 * @prefix  DATA
 *
 * @brief   Database configuration header
 *
 * Provides interfaces to database configuration
 *
```

```c
 */

#ifndef DATABASE_CFG_H_
#define DATABASE_CFG_H_

/*================== Includes =================================================*/
#include "general.h"

#include "batterysystem_cfg.h"

/*================== Macros and Definitions ==================================*/

/**
 * @brief maximum amount of data block
 *
 * this value is extendible but limitation is done due to RAM consumption and performance
 */
#define DATA_MAX_BLOCK_NR                 25        /* max 25 Blocks currently supported*/

/**
 * @brief data block identification number
 */
typedef enum {
    DATA_BLOCK_00       =  0,
    DATA_BLOCK_01       =  1,
    DATA_BLOCK_02       =  2,
    DATA_BLOCK_03       =  3,
    DATA_BLOCK_04       =  4,
    DATA_BLOCK_05       =  5,
    DATA_BLOCK_06       =  6,
    DATA_BLOCK_07       =  7,
    DATA_BLOCK_08       =  8,
    DATA_BLOCK_09       =  9,
    DATA_BLOCK_10       = 10,
    DATA_BLOCK_11       = 11,
    DATA_BLOCK_12       = 12,
    DATA_BLOCK_13       = 13,
    DATA_BLOCK_14       = 14,
    DATA_BLOCK_15       = 15,
    DATA_BLOCK_16       = 16,
    DATA_BLOCK_17       = 17,
    DATA_BLOCK_18       = 18,
    DATA_BLOCK_19       = 19,
    DATA_BLOCK_20       = 20,
    DATA_BLOCK_21       = 21,
    DATA_BLOCK_22       = 22,
    DATA_BLOCK_23       = 23,
    DATA_BLOCK_24       = 24,
    DATA_BLOCK_MAX      = DATA_MAX_BLOCK_NR,
} DATA_BLOCK_ID_TYPE_e;


```

```c
/**
 * @brief data block access types
 *
 * read or write access types
 */
typedef enum {
    WRITE_ACCESS = 0,   /*!< write access to data block */
    READ_ACCESS  = 1,   /*!< read access to data block  */
} DATA_BLOCK_ACCESS_TYPE_e;

/**
 * configuration struct of database channel (data block)
 */
typedef struct {
    void *blockptr;
    uint16_t datalength;
} DATA_BASE_HEADER_s;

/**
 * configuration struct of database device
 */
typedef struct {
    uint8_t nr_of_blockheader;
    DATA_BASE_HEADER_s *blockheaderptr;
} DATA_BASE_HEADER_DEV_s;


/**
 * Definitions for each database entry
 */
#define DATA_BLOCK_ID_CELLVOLTAGE                       DATA_BLOCK_00
#define DATA_BLOCK_ID_CELLTEMPERATURE                   DATA_BLOCK_01
#define DATA_BLOCK_ID_SOX                               DATA_BLOCK_02
#define DATA_BLOCK_ID_BALANCING_CONTROL_VALUES          DATA_BLOCK_03
#define DATA_BLOCK_ID_BALANCING_FEEDBACK_VALUES         DATA_BLOCK_04
#define DATA_BLOCK_ID_CURRENT_SENSOR                    DATA_BLOCK_05
#define DATA_BLOCK_ID_HW_INFO                           DATA_BLOCK_06
#define DATA_BLOCK_ID_STATEREQUEST                      DATA_BLOCK_07
#define DATA_BLOCK_ID_MINMAX                            DATA_BLOCK_08
#define DATA_BLOCK_ID_ISOGUARD                          DATA_BLOCK_09
#define DATA_BLOCK_ID_SLAVE_CONTROL                     DATA_BLOCK_10
#define DATA_BLOCK_ID_OPEN_WIRE                         DATA_BLOCK_11
#define DATA_BLOCK_ID_LTC_DEVICE_PARAMETER              DATA_BLOCK_12
#define DATA_BLOCK_ID_LTC_ACCURACY                      DATA_BLOCK_13
#define DATA_BLOCK_ID_ERRORSTATE                        DATA_BLOCK_14
#define DATA_BLOCK_ID_MSL                               DATA_BLOCK_15
#define DATA_BLOCK_ID_RSL                               DATA_BLOCK_16
#define DATA_BLOCK_ID_MOL                               DATA_BLOCK_17
#define DATA_BLOCK_ID_MOV_AVERAGE                       DATA_BLOCK_18
#define DATA_BLOCK_ID_CONTFEEDBACK                      DATA_BLOCK_19
#define DATA_BLOCK_ID_ILCKFEEDBACK                      DATA_BLOCK_20
#define DATA_BLOCK_ID_SYSTEMSTATE                       DATA_BLOCK_21
```

```c
#define DATA_BLOCK_ID_SOF                               DATA_BLOCK_22
#define DATA_BLOCK_ID_ALLGPIOVOLTAGE                    DATA_BLOCK_23
#define DATA_BLOCK_ID_CONT_SOH                          DATA_BLOCK_24

/**
 * data block struct of cell voltage
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                        */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry                   */
    uint16_t voltage[BS_NR_OF_BAT_CELLS];       /*!< unit: mV                                           */
    uint32_t valid_volt[BS_NR_OF_MODULES];      /*!< bitmask if voltages are valid. 0->valid, 1->invalid */
    uint32_t sumOfCells[BS_NR_OF_MODULES];      /*!< unit: mV                                           */
    uint8_t valid_socPECs[BS_NR_OF_MODULES];    /*!< 0 -> if PEC okay; 1 -> PEC error                   */
    uint32_t packVoltage_mV;                    /*!< uint: mV                                           */
    uint8_t state;                              /*!< for future use                                     */
} DATA_BLOCK_CELLVOLTAGE_s;

/**
 * data block struct of cell voltage
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                     /*!< timestamp of database entry                */
    uint32_t previous_timestamp;            /*!< timestamp of last database entry           */
    uint8_t openwire[BS_NR_OF_MODULES * (BS_NR_OF_BAT_CELLS_PER_MODULE+1)];   /*!< 1 -> open wire, 0 -> everything ok
    */
    uint8_t state;                          /*!< for future use                        */
} DATA_BLOCK_OPENWIRE_s;

/**
 * data block struct of cell temperatures
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock                           */
    uint32_t timestamp;                             /*!< timestamp of database entry                               */
    uint32_t previous_timestamp;                    /*!< timestamp of last database entry                          */
    int16_t temperature[BS_NR_OF_TEMP_SENSORS];     /*!< unit: degree Celsius                                      */
    uint16_t valid_temperature[BS_NR_OF_MODULES];   /*!< bitmask if temperatures are valid. 0->valid, 1->invalid */
    uint8_t state;                                  /*!< for future use                                            */
} DATA_BLOCK_CELLTEMPERATURE_s;

/**
 * data block struct of sox
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                 /*!< timestamp of database entry                */
    uint32_t previous_timestamp;        /*!< timestamp of last database entry           */
    float soc_mean;                     /*!< 0.0 <= soc_mean <= 100.0           */
    float soc_min;                      /*!< 0.0 <= soc_min <= 100.0            */
    float soc_max;                      /*!< 0.0 <= soc_max <= 100.0            */
    uint8_t state;                      /*!<                                    */
} DATA_BLOCK_SOX_s;


/**
 * data block struct of sof limits
 */
typedef struct {
    uint32_t timestamp;                     /*!< timestamp of database entry                        */
    uint32_t previous_timestamp;            /*!< timestamp of last database entry                   */
    float recommended_continuous_charge;    /*!< recommended continuous operating charge current    */
    float recommended_continuous_discharge; /*!< recommended continuous operating discharge current */
    float recommended_peak_charge;          /*!< recommended peak operating charge current          */
    float recommended_peak_discharge;       /*!< recommended peak operating discharge current       */
    float continuous_charge_MOL;            /*!< charge current maximum operating level             */
    float continuous_discharge_MOL;         /*!< discharge current maximum operating level          */
    float continuous_charge_RSL;            /*!< charge current recommended safety level            */
    float continuous_discharge_RSL;         /*!< discharge current recommended safety level         */
    float continuous_charge_MSL;            /*!< charge current maximum safety level                */
    float continuous_discharge_MSL;         /*!< discharge current maximum safety level             */
} DATA_BLOCK_SOF_s;


/**
 * data structure declaration of DATA_BLOCK_BALANCING_CONTROL
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint8_t balancing_state[BS_NR_OF_BAT_CELLS];    /*!< 0 means balancing is active, 0 means balancing is inactive*/
    uint32_t delta_charge[BS_NR_OF_BAT_CELLS];    /*!< Difference in Depth-of-Discharge in mAs*/
    uint8_t enable_balancing;           /*!< Switch for enabling/disabling balancing    */
    uint8_t threshold;                  /*!< balancing threshold in mV                  */
    uint8_t request;                     /*!< balancing request per CAN                 */
    uint8_t state;                      /*!< for future use                             */
} DATA_BLOCK_BALANCING_CONTROL_s;

/**
 * data structure declaration of DATA_BLOCK_USER_IO_CONTROL
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint8_t io_value_out[BS_NR_OF_MODULES];   /*!< data to be written to the port expander    */
    uint8_t io_value_in[BS_NR_OF_MODULES];    /*!< data read from to the port expander        */
    uint8_t eeprom_value_write[BS_NR_OF_MODULES];   /*!< data to be written to the slave EEPROM    */
    uint8_t eeprom_value_read[BS_NR_OF_MODULES];    /*!< data read from to the slave EEPROM        */
    uint8_t external_sensor_temperature[BS_NR_OF_MODULES];    /*!< temperature from the external sensor on slave   */
    uint32_t eeprom_read_address_to_use;                 /*!< address to read from for  slave EEPROM        */
    uint32_t eeprom_read_address_last_used;                  /*!< last address used to read fromfor slave EEPROM        */
    uint32_t eeprom_write_address_to_use;                  /*!< address to write to for slave EEPROM        */
    uint32_t eeprom_write_address_last_used;                  /*!< last address used to write to for slave EEPROM        */
    uint8_t state;                      /*!< for future use                             */
} DATA_BLOCK_SLAVE_CONTROL_s;

/**
 * data block struct of cell balancing feedback
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint16_t value[BS_NR_OF_MODULES];    /*!< unit: mV (opto-coupler output)     */
    uint8_t state;                      /*!< for future use                     */
} DATA_BLOCK_BALANCING_FEEDBACK_s;


/**
 * data block struct of user multiplexer values
 */

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint16_t value[BS_N_MUX_CHANNELS_PER_MUX*BS_N_USER_MUX_PER_LTC*BS_NR_OF_MODULES];               /*!< unit: mV (mux voltage input)       */
    uint8_t state;                                  /*!< for future use                     */
} DATA_BLOCK_USER_MUX_s;

/**
 * data block struct of current measurement
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    int32_t current;                                       /*!< unit: mA                */
    float voltage[BS_NR_OF_VOLTAGES_FROM_CURRENT_SENSOR];  /*!< unit: mV                */
    float temperature;                                     /*!< unit: 0.1&deg;C             */
    float power;                                           /*!< unit: W                */
    float current_counter;                                 /*!< unit: A.s               */
    float energy_counter;                                  /*!< unit: W.h               */
    uint8_t state_current;
    uint8_t state_voltage;
    uint8_t state_temperature;
    uint8_t state_power;
    uint8_t state_cc;
    uint8_t state_ec;
    uint8_t newCurrent;
    uint8_t newPower;
    uint32_t previous_timestamp_cur;                         /*!< timestamp of current database entry   */
    uint32_t timestamp_cur;                                  /*!< timestamp of current database entry   */
    uint32_t previous_timestamp_cc;                          /*!< timestamp of C-C database entry   */
    uint32_t timestamp_cc;                                   /*!< timestamp of C-C database entry   */
} DATA_BLOCK_CURRENT_SENSOR_s;

/**
 * data block struct of hardware info
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock        */
    uint32_t timestamp;                          /*!< timestamp of database entry                 */
    uint32_t previous_timestamp;                 /*!< timestamp of last database entry            */
    float vbat_mV;                               /*!< unit: mV                                    */
    float temperature;                           /*!< unit: degree Celsius                        */
    uint8_t state_vbat;                          /*!<                                             */
    uint8_t state_temperature;                   /*!<                                             */
} DATA_BLOCK_HW_INFO_s;

/**
 * data block struct of can state request
 */

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                          /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                 /*!< timestamp of last database entry           */
    uint8_t state_request;
    uint8_t previous_state_request;
    uint8_t state_request_pending;
    uint8_t state;
} DATA_BLOCK_STATEREQUEST_s;

/**
 * data block struct of LTC minimum and maximum values
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                          /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                 /*!< timestamp of last database entry           */
    uint32_t voltage_mean;
    uint16_t voltage_min;
    uint16_t voltage_module_number_min;
    uint16_t voltage_cell_number_min;
    uint16_t previous_voltage_min;
    uint16_t voltage_max;
    uint16_t voltage_module_number_max;
    uint16_t voltage_cell_number_max;
    uint16_t previous_voltage_max;
    float temperature_mean;
    int16_t temperature_min;
    uint16_t temperature_module_number_min;
    uint16_t temperature_sensor_number_min;
    int16_t temperature_max;
    uint16_t temperature_module_number_max;
    uint16_t temperature_sensor_number_max;
    uint8_t state;
} DATA_BLOCK_MINMAX_s;

/**
 * data block struct of isometer measurement
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;             /*!< timestamp of database entry                */
    uint32_t previous_timestamp;    /*!< timestamp of last database entry           */
    uint8_t valid;                  /*!< 0 -> valid, 1 -> resistance unreliable                              */
    uint8_t state;                  /*!< 0 -> resistance/measurement OK , 1 -> resistance too low or error   */
    uint32_t resistance_kOhm;       /*!< insulation resistance measured in kOhm                              */
} DATA_BLOCK_ISOMETER_s;

/**
 * data block struct of ltc device parameter
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint32_t sumOfCells[BS_NR_OF_MODULES];
    uint8_t valid_sumOfCells[BS_NR_OF_MODULES];          /*!< 0 -> valid, 1 ->
    unreliable                                         */
    uint16_t dieTemperature[BS_NR_OF_MODULES];           /* die temperature in degree
    celsius                                     */
    uint8_t valid_dieTemperature[BS_NR_OF_MODULES];      /*!< 0 -> valid, 1 ->
    unreliable                                         */
    uint32_t analogSupplyVolt[BS_NR_OF_MODULES];         /* voltage in
    [uV]                                                       */
    uint8_t valid_analogSupplyVolt[BS_NR_OF_MODULES];    /*!< 0 -> valid, 1 ->
    unreliable                                         */
    uint32_t digitalSupplyVolt[BS_NR_OF_MODULES];        /* voltage in
    [uV]                                                       */
    uint8_t valid_digitalSupplyVolt[BS_NR_OF_MODULES];   /*!< 0 -> valid, 1 ->
    unreliable                                         */
    uint32_t valid_cellvoltages[BS_NR_OF_MODULES];       /*!< 0 -> valid, 1 -> invalid, bit0 -> cell 0, bit1 -> cell 1
    ...       */
    uint8_t valid_GPIOs[BS_NR_OF_MODULES];               /*!< 0 -> valid, 1 -> invalid, bit0 -> GPIO0, bit1 -> GPIO1
    ...         */
    uint8_t valid_LTC[BS_NR_OF_MODULES];                 /*!< 0 -> LTC working, 1 -> LTC
    defect                                    */
} DATA_BLOCK_LTC_DEVICE_PARAMETER_s;

/**
```

```
 * data block struct of ltc adc accuracy measurement
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    int adc1_deviation[BS_NR_OF_MODULES];       /* ADC1 deviation from 2nd reference */
    int adc2_deviation[BS_NR_OF_MODULES];       /* ADC2 deviation from 2nd reference */
} DATA_BLOCK_LTC_ADC_ACCURACY_s;

/**
 * data block struct of error flags
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                              /*!< timestamp of database entry      */
    uint32_t previous_timestamp;                     /*!< timestamp of last database entry */
    uint8_t currentsensorresponding;                 /*!< 0 -> no error, 1 -> error        */
    uint8_t main_plus;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t main_minus;                              /*!< 0 -> no error, 1 -> error        */
    uint8_t precharge;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t charge_main_plus;                        /*!< 0 -> no error, 1 -> error        */
    uint8_t charge_main_minus;                       /*!< 0 -> no error, 1 -> error        */
    uint8_t charge_precharge;                        /*!< 0 -> no error, 1 -> error        */
    uint8_t interlock;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t crc_error;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t mux_error;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t spi_error;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t ltc_config_error;                        /*!< 0 -> no error, 1 -> error         */
    uint8_t insulation_error;                        /*!< 0 -> no error, 1 -> error        */
    uint8_t fuse_state_normal;                       /*!< 0 -> fuse ok,  1 -> fuse tripped */
    uint8_t fuse_state_charge;                       /*!< 0 -> fuse ok,  1 -> fuse tripped */
    uint8_t open_wire;                               /*!< 0 -> no error, 1 -> error        */
    uint8_t can_timing;                              /*!< 0 -> no error, 1 -> error        */
    uint8_t can_timing_cc;                           /*!< 0 -> no error, 1 -> error        */
    uint8_t mcuDieTemperature;                       /*!< 0 -> no error, 1 -> error        */
    uint8_t coinCellVoltage;                         /*!< 0 -> no error, 1 -> error        */
    uint8_t plausibilityCheck;                       /*!< 0 -> no error, else: error       */
    uint8_t deepDischargeDetected;                   /*!< 0 -> no error, 1 -> error        */
    uint8_t currentOnOpenPowerline;                  /*!< 0 -> no error, 1 -> error        */
} DATA_BLOCK_ERRORSTATE_s;

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock  */
    uint32_t timestamp;                     /*!< timestamp of database entry                */
    uint32_t previous_timestamp;            /*!< timestamp of last database entry           */
    uint8_t over_voltage;                   /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t under_voltage;                  /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_temperature_charge;        /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_temperature_discharge;     /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t under_temperature_charge;       /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t under_temperature_discharge;    /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
```

```c
    uint8_t over_current_charge_cell;       /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_current_charge_pl0;        /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_current_charge_pl1;        /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_current_discharge_cell;    /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_current_discharge_pl0;     /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t over_current_discharge_pl1;     /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t pcb_over_temperature;           /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
    uint8_t pcb_under_temperature;          /*!< 0 -> MSL NOT violated, 1 -> MSL violated   */
} DATA_BLOCK_MSL_FLAG_s;

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock  */
    uint32_t timestamp;                     /*!< timestamp of database entry                */
    uint32_t previous_timestamp;            /*!< timestamp of last database entry           */
    uint8_t over_voltage;                   /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t under_voltage;                  /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_temperature_charge;        /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_temperature_discharge;     /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t under_temperature_charge;       /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t under_temperature_discharge;    /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_charge_cell;       /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_charge_pl0;        /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_charge_pl1;        /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_discharge_cell;    /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_discharge_pl0;     /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t over_current_discharge_pl1;     /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t pcb_over_temperature;           /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
    uint8_t pcb_under_temperature;          /*!< 0 -> RSL NOT violated, 1 -> RSL violated   */
} DATA_BLOCK_RSL_FLAG_s;

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                     /*!< timestamp of database entry                */
    uint32_t previous_timestamp;            /*!< timestamp of last database entry           */
    uint8_t over_voltage;                   /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t under_voltage;                  /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_temperature_charge;        /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_temperature_discharge;     /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t under_temperature_charge;       /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t under_temperature_discharge;    /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_charge_cell;       /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_charge_pl0;        /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_charge_pl1;        /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_discharge_cell;    /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_discharge_pl0;     /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t over_current_discharge_pl1;     /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t pcb_over_temperature;           /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
    uint8_t pcb_under_temperature;          /*!< 0 -> MOL NOT violated, 1 -> MOL violated    */
} DATA_BLOCK_MOL_FLAG_s;

typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;               /*!< timestamp of database entry                */
    uint32_t previous_timestamp;      /*!< timestamp of last database entry           */
    float movAverage_current_1s;      /*!< current moving average over the last 1s               */
    float movAverage_current_5s;      /*!< current moving average over the last 5s               */
    float movAverage_current_10s;     /*!< current moving average over the last 10s              */
    float movAverage_current_30s;     /*!< current moving average over the last 30s              */
    float movAverage_current_60s;     /*!< current moving average over the last 60s              */
    float movAverage_current_config;  /*!< current moving average over the last configured time  */
    float movAverage_power_1s;        /*!< power moving average over the last 1s                 */
    float movAverage_power_5s;        /*!< power moving average over the last 5s                 */
    float movAverage_power_10s;       /*!< power moving average over the last 10s                */
    float movAverage_power_30s;       /*!< power moving average over the last 30s                */
    float movAverage_power_60s;       /*!< power moving average over the last 60s                */
    float movAverage_power_config;    /*!< power moving average over the last configured time    */
} DATA_BLOCK_MOVING_AVERAGE_s;

/**
 * data block struct of contactor feedback
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint16_t contactor_feedback;                /*!< feedback of contactors, without interlock */
} DATA_BLOCK_CONTFEEDBACK_s;

/**
 * data block struct of interlock feedback
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint8_t interlock_feedback;                 /*!< feedback of interlock, without contactors */
} DATA_BLOCK_ILCKFEEDBACK_s;

/**
 * data block struct of system state
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                         /*!< timestamp of database entry                */
    uint32_t previous_timestamp;                /*!< timestamp of last database entry           */
    uint8_t bms_state;                          /*!< system state (e.g., standby, normal) */
} DATA_BLOCK_SYSTEMSTATE_s;

/**
 * data block struct of cell voltage
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;                                                  /*!< timestamp of database
    entry                         */
    uint32_t previous_timestamp;                                          /*!< timestamp of last database
    entry                    */
    uint16_t gpiovoltage[BS_NR_OF_MODULES * BS_NR_OF_GPIOS_PER_MODULE];  /*!< unit:
    mV                                            */
    uint16_t valid_gpiovoltages[BS_NR_OF_MODULES];                        /*!< bitmask if voltages are valid.
    0->valid, 1->invalid */
    uint8_t state;                                                        /*!< for future
    use                                       */
} DATA_BLOCK_ALLGPIOVOLTAGE_s;

/**
 * data block struct of contactor SOH
 */
typedef struct {
    /* Timestamp info needs to be at the beginning. Automatically written on DB_WriteBlock */
    uint32_t timestamp;  /*!< timestamp of database entry */
    uint32_t previous_timestamp;  /*!< timestamp of last database entry */
    float contactor_soh[BS_NR_OF_CONTACTORS];  /*!< SOH of contactors */
} DATA_BLOCK_CONT_SOH_s;

/*================== Extern Constant and Variable Declarations ==============*/

/**
 * @brief device configuration of database
 *
 * all attributes of device configuration are listed here (pointer to channel list, number of channels)
 */
extern const DATA_BASE_HEADER_DEV_s data_base_dev;

/*================== Extern Function Prototypes ==============================*/

#endif /* DATABASE_CFG_H_ */
```