```
/**
 *
 * @copyright &copy; 2010 - 2020, Fraunhofer-Gesellschaft zur Foerderung der
 *  angewandten Forschung e.V. All rights reserved.
 *
 * BSD 3-Clause License
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 * 2.  Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 * 3.  Neither the name of the copyright holder nor the names of its
 *     contributors may be used to endorse or promote products derived from
 *     this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 * We kindly request you to use one or more of the following phrases to refer
 * to foxBMS in your hardware, software, documentation or advertising
 * materials:
 *
 * &Prime;This product uses parts of foxBMS&reg;&Prime;
 *
 * &Prime;This product includes parts of foxBMS&reg;&Prime;
 *
 * &Prime;This product is derived from foxBMS&reg;&Prime;
 *
 */

/**
 * @file    diag_cfg.h
 * @author  foxBMS Team
 * @date    09.11.2015 (date of creation)
 * @ingroup ENGINE_CONF
 * @prefix  DIAG
 *
 * @brief   Diagnostic module configuration header
 *
 * In this header filer are the different diagnosis channel
 * defines assigned to different diagnosis IDs.
```

```
 *
 * Furthermore are the diagnosis error log settings be configured here..
 */

#ifndef DIAG_CFG_H_
#define DIAG_CFG_H_

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

#include "batterysystem_cfg.h"

/*================== Macros and Definitions ====================================*/
#define DIAG_ERROR_SENSITIVITY_HIGH         (0)    /* logging at first event */
#define DIAG_ERROR_SENSITIVITY_MID          (5)    /* logging at fifth event */
#define DIAG_ERROR_SENSITIVITY_LOW          (10)   /* logging at tenth event */

#define DIAG_ERROR_VOLTAGE_SENSITIVITY_MSL      (500)   /*!< MSL level for event occurrence if over/under voltage event   */
#define DIAG_ERROR_VOLTAGE_SENSITIVITY_RSL      (500)   /*!< RSL level for event occurrence if over/under voltage event   */
#define DIAG_ERROR_VOLTAGE_SENSITIVITY_MOL      (500)   /*!< MOL level for event occurrence if over/under voltage event   */

#define DIAG_ERROR_TEMPERATURE_SENSITIVITY_MSL  (500)   /*!< MSL level for event occurrence if over/under temperature event    */
#define DIAG_ERROR_TEMPERATURE_SENSITIVITY_RSL  (500)   /*!< RSL level for event occurrence if over/under temperature event    */
#define DIAG_ERROR_TEMPERATURE_SENSITIVITY_MOL  (500)   /*!< MOL level for event occurrence if over/under temperature event    */

#define DIAG_ERROR_CURRENT_SENSITIVITY_MSL      (500)   /*!< MSL level for event occurrence if over/under current event        */
#define DIAG_ERROR_CURRENT_SENSITIVITY_RSL      (500)   /*!< RSL level for event occurrence if over/under current event        */
#define DIAG_ERROR_CURRENT_SENSITIVITY_MOL      (500)   /*!< MOL level for event occurrence if over/under current event        */

#define DIAG_ERROR_SLAVE_TEMP_SENSITIVITY_MSL   (500)   /*!< MSL level for event occurrence if slave PCB temperature event     */
#define DIAG_ERROR_SLAVE_TEMP_SENSITIVITY_RSL   (500)   /*!< RSL level for event occurrence if slave PCB temperature event     */
#define DIAG_ERROR_SLAVE_TEMP_SENSITIVITY_MOL   (500)   /*!< MOL level for event occurrence if slave PCB temperature event     */

#define DIAG_ERROR_LTC_PEC_SENSITIVITY          (5)
#define DIAG_ERROR_LTC_MUX_SENSITIVITY          (5)
#define DIAG_ERROR_LTC_SPI_SENSITIVITY          (5)

#define DIAG_ERROR_CAN_TIMING_SENSITIVITY       (100)
#define DIAG_ERROR_CAN_TIMING_CC_SENSITIVITY    (100)
#define DIAG_ERROR_CAN_SENSOR_SENSITIVITY       (100)
```

```
93
94  #define DIAG_ERROR_MAIN_PLUS_SENSITIVITY        (50)
95  #define DIAG_ERROR_MAIN_MINUS_SENSITIVITY       (50)
96  #define DIAG_ERROR_PRECHARGE_SENSITIVITY        (50)
97
98  #define DIAG_ERROR_INTERLOCK_SENSITIVITY        (10)
99
100 #define DIAG_ERROR_INSULATION_SENSITIVITY       (30)
101
102 #define DIAG_ERROR_PLAUSIBILITY_PACK_SENSITIVITY    (100)
103
104 /** Number of errors that can be logged */
105 #define DIAG_FAIL_ENTRY_LENGTH              (50)
106
107 /** Maximum number of the same errors that are logged */
108 #define DIAG_MAX_ENTRIES_OF_ERROR           (5)
109
110 /** Number of contactor errors that are logged */
111 #define DIAG_FAIL_ENTRY_CONTACTOR_LENGTH    (50)
112
113
114 typedef enum {
115     DIAG_CH_FLASHCHECKSUM,                          /*  */
116     DIAG_CH_BKPDIAG_FAILURE,                        /*  */
117     DIAG_CH_WATCHDOGRESET_FAILURE,                  /*  */
118     DIAG_CH_POSTOSINIT_FAILURE,                     /*  */
119     DIAG_CH_CALIB_EEPR_FAILURE,                     /*  */
120     DIAG_CH_CAN_INIT_FAILURE,                       /*  */
121     DIAG_CH_VIC_INIT_FAILURE,
122     /* HW-/SW-Runtime events: 16-31 */
123     DIAG_CH_DIV_BY_ZERO_FAILURE,                    /*  */
124     DIAG_CH_UNDEF_INSTRUCTION_FAILURE,              /*  */
125     DIAG_CH_DATA_BUS_FAILURE,                       /*  */
126     DIAG_CH_INSTRUCTION_BUS_FAILURE,                /*  */
127     DIAG_CH_HARDFAULT_NOTHANDLED,                   /*  */
128     DIAG_CH_RUNTIME_ERROR_RESERVED_1,               /*  reserved for future needs */
129     DIAG_CH_RUNTIME_ERROR_RESERVED_2,               /*  reserved for future needs */
130     DIAG_CH_RUNTIME_ERROR_RESERVED_3,               /*  reserved for future needs */
131     DIAG_CH_CONFIGASSERT,                           /*  */
132     DIAG_CH_SYSTEMMONITORING_TIMEOUT,               /*  */
133     /* Measurement events: 32-47 */
134     DIAG_CH_CANS_MAX_VALUE_VIOLATE,
135     DIAG_CH_CANS_MIN_VALUE_VIOLATE,
136     DIAG_CH_CANS_CAN_MOD_FAILURE,
137     DIAG_CH_ISOMETER_TIM_ERROR,                     /* Measured frequency too low or no new value captured during
        last cycle */
138     DIAG_CH_ISOMETER_GROUNDERROR,                   /* Ground error detected */
139     DIAG_CH_ISOMETER_ERROR,                         /* Device error, invalid measurement result */
140     DIAG_CH_ISOMETER_MEAS_INVALID,                  /* Measurement trustworthy or not, hysteresis to ground error
        flag */
141     DIAG_CH_CELLVOLTAGE_OVERVOLTAGE_MSL,            /* Cell voltage limits violated */
142     DIAG_CH_CELLVOLTAGE_OVERVOLTAGE_RSL,            /* Cell voltage limits violated */
```

```
143       DIAG_CH_CELLVOLTAGE_OVERVOLTAGE_MOL,            /* Cell voltage limits violated */
144       DIAG_CH_CELLVOLTAGE_UNDERVOLTAGE_MSL,           /* Cell voltage limits violated */
145       DIAG_CH_CELLVOLTAGE_UNDERVOLTAGE_RSL,           /* Cell voltage limits violated */
146       DIAG_CH_CELLVOLTAGE_UNDERVOLTAGE_MOL,           /* Cell voltage limits violated */
147       DIAG_CH_TEMP_OVERTEMPERATURE_CHARGE_MSL,        /* Temperature limits violated */
148       DIAG_CH_TEMP_OVERTEMPERATURE_CHARGE_RSL,        /* Temperature limits violated */
149       DIAG_CH_TEMP_OVERTEMPERATURE_CHARGE_MOL,        /* Temperature limits violated */
150       DIAG_CH_TEMP_OVERTEMPERATURE_DISCHARGE_MSL,     /* Temperature limits violated */
151       DIAG_CH_TEMP_OVERTEMPERATURE_DISCHARGE_RSL,     /* Temperature limits violated */
152       DIAG_CH_TEMP_OVERTEMPERATURE_DISCHARGE_MOL,     /* Temperature limits violated */
153       DIAG_CH_TEMP_UNDERTEMPERATURE_CHARGE_MSL,       /* Temperature limits violated */
154       DIAG_CH_TEMP_UNDERTEMPERATURE_CHARGE_RSL,       /* Temperature limits violated */
155       DIAG_CH_TEMP_UNDERTEMPERATURE_CHARGE_MOL,       /* Temperature limits violated */
156       DIAG_CH_TEMP_UNDERTEMPERATURE_DISCHARGE_MSL,    /* Temperature limits violated */
157       DIAG_CH_TEMP_UNDERTEMPERATURE_DISCHARGE_RSL,    /* Temperature limits violated */
158       DIAG_CH_TEMP_UNDERTEMPERATURE_DISCHARGE_MOL,    /* Temperature limits violated */
159       DIAG_CH_OVERCURRENT_CHARGE_CELL_MSL,            /* Overcurrent */
160       DIAG_CH_OVERCURRENT_CHARGE_CELL_RSL,            /* Overcurrent */
161       DIAG_CH_OVERCURRENT_CHARGE_CELL_MOL,            /* Overcurrent */
162       DIAG_CH_OVERCURRENT_DISCHARGE_CELL_MSL,         /* Overcurrent */
163       DIAG_CH_OVERCURRENT_DISCHARGE_CELL_RSL,         /* Overcurrent */
164       DIAG_CH_OVERCURRENT_DISCHARGE_CELL_MOL,         /* Overcurrent */
165       DIAG_CH_OVERCURRENT_CHARGE_PL0_MSL,             /* Overcurrent */
166       DIAG_CH_OVERCURRENT_CHARGE_PL0_RSL,             /* Overcurrent */
167       DIAG_CH_OVERCURRENT_CHARGE_PL0_MOL,             /* Overcurrent */
168       DIAG_CH_OVERCURRENT_CHARGE_PL1_MSL,             /* Overcurrent */
169       DIAG_CH_OVERCURRENT_CHARGE_PL1_RSL,             /* Overcurrent */
170       DIAG_CH_OVERCURRENT_CHARGE_PL1_MOL,             /* Overcurrent */
171       DIAG_CH_OVERCURRENT_DISCHARGE_PL0_MSL,          /* Overcurrent */
172       DIAG_CH_OVERCURRENT_DISCHARGE_PL0_RSL,          /* Overcurrent */
173       DIAG_CH_OVERCURRENT_DISCHARGE_PL0_MOL,          /* Overcurrent */
174       DIAG_CH_OVERCURRENT_DISCHARGE_PL1_MSL,          /* Overcurrent */
175       DIAG_CH_OVERCURRENT_DISCHARGE_PL1_RSL,          /* Overcurrent */
176       DIAG_CH_OVERCURRENT_DISCHARGE_PL1_MOL,          /* Overcurrent */
177       DIAG_CH_OVERCURRENT_PL_NONE,
178       DIAG_CH_LTC_SPI,                                /* LTC */
179       DIAG_CH_LTC_PEC,                                /* LTC */
180       DIAG_CH_LTC_MUX,                                /* LTC */
181       DIAG_CH_LTC_CONFIG,                             /* LTC */
182
183       /* Communication events: 50-63*/
184       DIAG_CH_CAN_TIMING,  /* error in CAN timing */
185       DIAG_CH_CAN_CC_RESPONDING, /* CAN C-C */
186       DIAG_CH_CURRENT_SENSOR_RESPONDING, /* Current sensor not responding anymore */
187       /* Contactor events: 69-77*/
188       DIAG_CH_CONTACTOR_DAMAGED, /* Opening contactor at over current */
189       DIAG_CH_CONTACTOR_OPENING, /* counter for contactor opening */
190       DIAG_CH_CONTACTOR_CLOSING, /* counter for contactor closing */
191       DIAG_CH_CONTACTOR_MAIN_PLUS_FEEDBACK, /* Contactor feedback error */
192       DIAG_CH_CONTACTOR_MAIN_MINUS_FEEDBACK, /* Contactor feedback error */
193       DIAG_CH_CONTACTOR_PRECHARGE_FEEDBACK, /* Contactor feedback error */
194       DIAG_CH_CONTACTOR_CHARGE_MAIN_PLUS_FEEDBACK, /* Contactor feedback error */
```

```
195       DIAG_CH_CONTACTOR_CHARGE_MAIN_MINUS_FEEDBACK, /* Contactor feedback error */
196       DIAG_CH_CONTACTOR_CHARGE_PRECHARGE_FEEDBACK, /* Contactor feedback error */
197       DIAG_CH_INTERLOCK_FEEDBACK, /* Interlock feedback error */
198       DIAG_CH_SLAVE_PCB_UNDERTEMPERATURE_MSL,
199       DIAG_CH_SLAVE_PCB_UNDERTEMPERATURE_RSL,
200       DIAG_CH_SLAVE_PCB_UNDERTEMPERATURE_MOL,
201       DIAG_CH_SLAVE_PCB_OVERTEMPERATURE_MSL,
202       DIAG_CH_SLAVE_PCB_OVERTEMPERATURE_RSL,
203       DIAG_CH_SLAVE_PCB_OVERTEMPERATURE_MOL,
204       DIAG_CH_INSULATION_ERROR, /* Insulation error: measured insulation < threshold */
205       DIAG_CH_FUSE_STATE_NORMAL, /* Fuse tripped */
206       DIAG_CH_FUSE_STATE_CHARGE, /* Fuse tripped */
207       DIAG_CH_ERROR_MCU_DIE_TEMPERATURE, /* MCU die temperature */
208       DIAG_CH_LOW_COIN_CELL_VOLTAGE, /* coin cell voltage */
209       DIAG_CH_CRIT_LOW_COIN_CELL_VOLTAGE, /* coin cell voltage */
210       DIAG_CH_OPEN_WIRE, /* open-wire check */
211       DIAG_CH_PLAUSIBILITY_CELL_VOLTAGE, /* plausibility checks */
212       DIAG_CH_PLAUSIBILITY_CELL_TEMP, /* plausibility checks */
213       DIAG_CH_PLAUSIBILITY_PACK_VOLTAGE, /* plausibility checks */
214       DIAG_CH_DEEP_DISCHARGE_DETECTED, /* DoD was detected */
215       DIAG_ID_MAX, /* MAX indicator - do not change */
216   } DIAG_CH_ID_e;
217
218   /** diagnosis check result (event) */
219   typedef enum {
220       DIAG_EVENT_OK, /*!< diag channel event OK */
221       DIAG_EVENT_NOK, /*!< diag channel event NOK */
222       DIAG_EVENT_RESET, /*!< reset diag channel eventcounter to 0 */
223   } DIAG_EVENT_e;
224
225   /**
226    * enable state of diagnosis entry
227    */
228   typedef enum {
229       DIAG_ENABLED,
230       DIAG_DISABLED,
231   } DIAG_ENABLE_STATE_e;
232
233
234   #if CHECK_CAN_TIMING == TRUE
235       #define DIAG_CAN_TIMING DIAG_ENABLED
236   #else
237       #define DIAG_CAN_TIMING DIAG_DISABLED
238   #endif
239
240   #if CURRENT_SENSOR_PRESENT == TRUE
241       #define DIAG_CAN_SENSOR_PRESENT DIAG_ENABLED
242   #else
243       #define DIAG_CAN_SENSOR_PRESENT DIAG_DISABLED
244   #endif
245
246   /**
```

```
 * diagnosis recording activation
 */
typedef enum {
    DIAG_RECORDING_ENABLED,     /*!< enable diagnosis event recording   */
    DIAG_RECORDING_DISABLED,    /*!< disable diagnosis event recording  */
} DIAG_TYPE_RECORDING_e;

/*  FIXME some enums are typedefed with DIAG...TYPE_e, some with DIAG_TYPE..._e! Reconsider this */
/**
 * diagnosis types for system monitoring
 */
typedef enum {
    DIAG_SYSMON_CYCLICTASK, /*!< */
    DIAG_SYSMON_RESERVED,   /*!< */
} DIAG_SYSMON_TYPE_e;

/**
 * diagnosis handling type for system monitoring
 */
typedef enum {
    DIAG_SYSMON_HANDLING_DONOTHING,             /*!< */
    DIAG_SYSMON_HANDLING_SWITCHOFFCONTACTOR,    /*!< */
} DIAG_SYSMON_HANDLING_TYPE_e;


/**
 * @brief listing of system-relevant tasks or functions which are checked by system monitoring
 *
 * diag_sysmon_ch_cfg[]=
 */
typedef enum {
    DIAG_SYSMON_DATABASE_ID,        /*!< diag entry for database               */
    DIAG_SYSMON_SYS_ID,             /*!< diag entry for sys                    */
    DIAG_SYSMON_BMS_ID,             /*!< diag entry for bms                    */
    DIAG_SYSMON_CONT_ID,            /*!< diag entry for contactors             */
    DIAG_SYSMON_ILCK_ID,            /*!< diag entry for contactors             */
    DIAG_SYSMON_LTC_ID,             /*!< diag entry for ltc                    */
    DIAG_SYSMON_ISOGUARD_ID,        /*!< diag entry for isoguard               */
    DIAG_SYSMON_CANS_ID,            /*!< diag entry for can                    */
    DIAG_SYSMON_APPL_CYCLIC_1ms,    /*!< diag entry for application 10ms task  */
    DIAG_SYSMON_APPL_CYCLIC_10ms,   /*!< diag entry for application 10ms task  */
    DIAG_SYSMON_APPL_CYCLIC_100ms,  /*!< diag entry for application 100ms task */
    DIAG_SYSMON_MODULE_ID_MAX,      /*!< end marker do not delete              */
} DIAG_SYSMON_MODULE_ID_e;

/*  FIXME doxygen comment */
/*  FIXME is DIAG_CODE_s an appropriate name for this? */
typedef struct {
    uint32_t GENERALmsk;
    uint32_t CELLMONmsk;
    uint32_t COMmsk;
    uint32_t ADCmsk;
```

```
299 } DIAG_CODE_s;
300
301
302 /**
303  * Channel configuration of one diag channel
304 */
305 typedef struct {
306     DIAG_CH_ID_e id;                        /*!< diagnosis event id diag_id */
307     uint8_t description[40];
308     uint16_t thresholds;                     /*!< threshold for number of events which will be tolerated before
    generating a notification in both direction (OK or NOT OK)
309                                              *   threshold = 0: reports the value at first occurence, threshold =
                                                 1:reports the value at second occurence*/
310     DIAG_TYPE_RECORDING_e enablerecording;  /*!< if enabled recording in diag_memory will be activated */
311     DIAG_ENABLE_STATE_e state;              /*!< if enabled diagnosis event will be evaluated */
312     void (*callbackfunc)(DIAG_CH_ID_e, DIAG_EVENT_e);     /*!< will be called if number of events exceeds threshold
    (in both direction) with parameter DIAG_EVENT_e */
313 } DIAG_CH_CFG_s;
314
315
316 /**
317  * struct for device Configuration of diag module
318  */
319 typedef struct {
320     uint8_t nr_of_ch;       /*!< number of entries in DIAG_CH_CFG_s */
321     DIAG_CH_CFG_s *ch_cfg;  /*!< pointer to diag channel config struct */
322 } DIAG_DEV_s;
323
324 /**
325  * state (in summary) used for task or function notification
326  */
327 typedef struct {
328     uint32_t state;     /*!< state              */
329     uint32_t timestamp; /*!< timestamp of state */
330 } DIAG_SYSMON_NOTIFICATION_s;
331
332
333 /**
334  * Channel configuration of one system monitoring channel
335  */
336 typedef struct {
337     DIAG_SYSMON_MODULE_ID_e id;                     /*!< the diag type by its symbolic name           */
338     DIAG_SYSMON_TYPE_e type;                        /*!< system monitoring types: cyclic or special   */
339     uint16_t threshold;                             /*!< max. delay time in ms                        */
340     DIAG_TYPE_RECORDING_e enablerecording;          /*!< enabled if set to DIAG_RECORDING_ENABLED     */
341     DIAG_SYSMON_HANDLING_TYPE_e handlingtype;       /*!< type of handling of system monitoring errors */
342     DIAG_ENABLE_STATE_e state;                      /*!< enable or disable system monitoring          */
343     void (*callbackfunc)(DIAG_SYSMON_MODULE_ID_e);  /*!< */
344 } DIAG_SYSMON_CH_CFG_s;
345
346 /*================== Extern Constant and Variable Declarations ==============*/
347 /**
```

```
348    * diag device configuration struct
349    */
350   extern DIAG_DEV_s diag_dev;
351
352   /**
353    * diag system monitoring struct
354    */
355   extern DIAG_SYSMON_CH_CFG_s diag_sysmon_ch_cfg[];
356   extern DIAG_CH_CFG_s  diag_ch_cfg[];
357
358   /*  FIXME why is it in header at all? and why is it in code at all? not used */
359   extern DIAG_CODE_s diag_mask;
360
361   /*================== Extern Function Prototypes ==============================*/
362   /**
363    * @brief update function for diagnosis flags (errors, MOL/RSL/MSL violations)
364    */
365   extern void DIAG_updateFlags(void);
366
367   #endif /* DIAG_CFG_H_ */
368
```