```
/**
 *
 * @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    can_cfg.h
 * @author  foxBMS Team
 * @date    12.07.2015 (date of creation)
 * @ingroup DRIVERS_CONF
 * @prefix  CAN
 *
 * @brief   Headers for the configuration for the CAN module
 *
 * The activation and the length of the message buffers as well as the number of
 * the messages that are received are to be configured here
```

```
 *
 */

#ifndef CAN_CFG_H_
#define CAN_CFG_H_

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

#include "cpu_cfg.h"
#include "io.h"

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

#define CAN_0_TRANS_STANDBY_CONTROL    IO_PIN_CAN_0_TRANS_STANDBY_CONTROL
#define CAN_1_TRANS_STANDBY_CONTROL    IO_PIN_CAN_1_TRANS_STANDBY_CONTROL

/* ****************************************************************************
 *  CAN BUFFER OPTIONS
 ****************************************************************************/
/**
 * @ingroup CONFIG_CAN
 * CAN0 bus baudrate. CAN peripheral prescaler and time quantums on microcontroller
 * will be configured accordingly. See STM32 Reference Manual p. 1097 for more
 *
 * \par Type:
 * select(4)
 * \par Default:
 * 1
*/

/**
 * defines the BAUD rate of the CAN0
 */
/* #define CAN0_BAUDRATE (1000000U) */
#define CAN0_BAUDRATE (500000U)
/* #define CAN0_BAUDRATE (250000U) */
/* #define CAN0_BAUDRATE (125000U) */

/**
 * @ingroup CONFIG_CAN
 * CAN1 bus baudrate. CAN peripheral prescaler and time quantums on microcontroller
 * will be configured accordingly. See STM32 Reference Manual p. 1097 for more
 *
 * \par Type:
 * select(4)
 * \par Default:
 * 1
*/

/**
 * defines the BAUD rate of the CAN1
```

```
105    */
106   /* #define CAN1_BAUDRATE (1000000U) */
107   #define CAN1_BAUDRATE (500000U)
108   /* #define CAN1_BAUDRATE (250000U) */
109   /* #define CAN1_BAUDRATE (125000U) */
110
111
112
113   /* CAN enabling */
114   /**
115    * @ingroup CONFIG_CAN
116    * Enables or disables CAN0
117    * \par Type:
118    * int
119    * \par Range:
120    * x == 0 or x == 1
121    * \par Default:
122    * 1
123   */
124   #define CAN_USE_CAN_NODE0                (1U)
125
126   /**
127    * @ingroup CONFIG_CAN
128    * Enables or disables CAN1
129    * \par Type:
130    * int
131    * \par Range:
132    * x == 0 or x == 1
133    * \par Default:
134    * 1
135   */
136   #define CAN_USE_CAN_NODE1                (1U)           CAN 1 is enabled.
137
138   /**
139    * @ingroup CONFIG_CAN
140    * Enables or disables transmitter standby control
141    * \par Type:
142    * int
143    * \par Range:
144    * x == 0 or x == 1
145    * \par Default:
146    * 1
147   */
148   #define CAN_USE_STANDBY_CONTROL          (1U)
149
150   /* transmit buffer */
151   /**
152    * @ingroup CONFIG_CAN
153    * Enables or disables CAN0 transmit buffer
154    * \par Type:
155    * int
156    * \par Range:
```

```
157   * x == 0 or x == 1
158   * \par Default:
159   * 1
160   */
161  #define CAN0_USE_TRANSMIT_BUFFER         (1U)
162  /**
163   * @ingroup CONFIG_CAN
164   * Defines CAN0 transmit buffer length
165   * \par Type:
166   * int
167   * \par Range:
168   * 0 < x
169   * \par Default:
170   * 16
171   */
172  #define CAN0_TRANSMIT_BUFFER_LENGTH      (24U)
173
174  /**
175   * @ingroup CONFIG_CAN
176   * Enables or disables CAN1 transmit buffer
177   * \par Type:
178   * int
179   * \par Range:
180   * x == 0 or x == 1
181   * \par Default:
182   * 1
183   */
184  #define CAN1_USE_TRANSMIT_BUFFER         (1U)
185  /**
186   * @ingroup CONFIG_CAN
187   * Defines CAN1 transmit buffer length
188   * \par Type:
189   * int
190   * \par Range:
191   * 0 < x
192   * \par Default:
193   * 16
194   */
195  #define CAN1_TRANSMIT_BUFFER_LENGTH      (16U)
196
197  /* receive buffer */
198  /**
199   * @ingroup CONFIG_CAN
200   * Enables or disables CAN0 receive buffer
201   * \par Type:
202   * int
203   * \par Range:
204   * x == 0 or x == 1
205   * \par Default:
206   * 1
207   */
208  #define CAN0_USE_RECEIVE_BUFFER          (1U)
```

```
209  /**
210   * @ingroup CONFIG_CAN
211   * Defines CAN0 receive buffer length
212   * \par Type:
213   * int
214   * \par Range:
215   * 0 < x
216   * \par Default:
217   * 16
218  */
219  #define CAN0_RECEIVE_BUFFER_LENGTH       (16U)
220
221  /**
222   * @ingroup CONFIG_CAN
223   * Enables or disables CAN1 receive buffer
224   * \par Type:
225   * int
226   * \par Range:
227   * x == 0 or x == 1
228   * \par Default:
229   * 1
230  */
231  #define CAN1_USE_RECEIVE_BUFFER          (1U)
232  /**
233   * @ingroup CONFIG_CAN
234   * Defines CAN1 receive buffer length
235   * \par Type:
236   * int
237   * \par Range:
238   * 0 < x
239   * \par Default:
240   * 16
241  */
242  #define CAN1_RECEIVE_BUFFER_LENGTH       (16U)
243
244
245  /* Number of messages that will bypass the receive buffer and will be interpreted right on reception.
246   * Set the respective IDs and implement the wished functionality either in individual callback
247   * function or in default STD_RETURN_TYPE_e CAN_BufferBypass(CAN_NodeTypeDef_e canNode, uint32_t msgID,
248   * uint8_t* data, uint8_t DLC, uint8_t RTR) function in the can.c file. Use bypassing only for
249   * important messages because of handling during ISR */
250  /**
251   * @ingroup CONFIG_CAN
252   * Defines number of RX messages that bypass receive buffer on CAN0 bus
253   * \par Type:
254   * int
255   * \par Range:
256   * 0 <= x
257   * \par Default:
258   * 0
259  */
260  #define CAN0_BUFFER_BYPASS_NUMBER_OF_IDs (1U)
```

```
261
262 /**
263  * @ingroup CONFIG_CAN
264  * Defines number of RX messages that bypass receive buffer on CAN1 bus
265  * \par Type:
266  * int
267  * \par Range:
268  * 0 <= x
269  * \par Default:
270  * 0
271 */
272 #define CAN1_BUFFER_BYPASS_NUMBER_OF_IDs (0U)
273
274 /**
275  * @ingroup CONFIG_CAN
276  * Defines CAN message ID to perform a software reset
277  * \par Type:
278  * int
279  * \par Default:
280  * 351
281 */
282 #define CAN_ID_SOFTWARE_RESET_MSG                    (0x95U)
283
284 /**
285  * @ingroup CONFIG_CAN
286  * When enabled unique device ID is need as can data to perform SW reset
287  * \par Type:
288  * int
289  * \par Range:
290  * [0,1]
291  * \par Default:
292  * 0
293 */
294 /* #define CAN_SW_RESET_WITH_DEVICE_ID        (01) */
295 #define CAN_SW_RESET_WITH_DEVICE_ID        (0U)        Any device can reset foxBMS.
296
297 typedef struct CAN_MSG_RX_TYPE {
298     uint32_t ID;    /*!< message ID */
299     uint32_t mask;  /*!< mask or 0x0000 to select list mode */
300     uint8_t DLC;    /*!< data length */
301     uint8_t RTR;    /*!< rtr bit */
302     uint32_t fifo;  /*!< selected CAN hardware (CAN_FILTER_FIFO0 or CAN_FILTER_FIFO1) */
303     STD_RETURN_TYPE_e (*func)(uint32_t ID, uint8_t*, uint8_t, uint8_t);  /*!< callback function */
304 } CAN_MSG_RX_TYPE_s;
305
306
307 typedef uint32_t (*can_callback_funcPtr)(uint32_t idx, void * value);
308
309 /**
310  * type definition for structure of a CAN message with its
311  *  ID,
312  *  data length code,
```

Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)

```
 *  repetition rate (stated in number of calls of CANS mainfunction = ticks),
 *  the initial phase,
 *  a callback function if transfer of TX message to CAN module is successful
 */
typedef struct  {
    uint32_t ID;                    /*!< CAN message id */
    uint8_t DLC;                    /*!< CAN message data length code */
    uint32_t repetition_time;       /*!< CAN message cycle time */
    uint32_t repetition_phase;      /*!< CAN message startup (first send) offset */
    can_callback_funcPtr cbk_func;  /*!< CAN message callback after message is sent or received */
} CAN_MSG_TX_TYPE_s;

typedef struct CanPdu {          PDU stands for Protocol Data Unit (i.e. Message Format).
    uint8_t sdu[8];              SDU stands for Service Data Unit
    uint32_t id;
    uint8_t dlc;
} Can_PduType;


/*================== Constant and Variable Definitions ====================*/
extern CAN_HandleTypeDef hcan0;
extern CAN_HandleTypeDef hcan1;
extern CAN_MSG_RX_TYPE_s can0_RxMsgs[];
extern CAN_MSG_RX_TYPE_s can1_RxMsgs[];
extern uint32_t can0_bufferBypass_RxMsgs[CAN0_BUFFER_BYPASS_NUMBER_OF_IDs];
extern uint32_t can1_bufferBypass_RxMsgs[CAN1_BUFFER_BYPASS_NUMBER_OF_IDs];
extern const CAN_MSG_TX_TYPE_s can_CAN0_messages_tx[];
extern const CAN_MSG_TX_TYPE_s can_CAN1_messages_tx[];


/**
 * array length for receiving CAN0 message definition
 */
extern const uint8_t can_CAN0_rx_length;

/**
 * array length for receiving CAN1 message definition
 */
extern const uint8_t can_CAN1_rx_length;

/**
 * array length for transmission CAN0 message definition
 */
extern const uint8_t can_CAN0_tx_length;

/**
 * array length for transmission CAN1 message definition
 */
extern const uint8_t can_CAN1_tx_length;

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

```

```
365
366   /*================== Function Implementations ============================*/
367
368   #endif /* CAN_CFG_H_ */
369
```