```
/**
 *
 * @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    cansignal_cfg.c
 * @author  foxBMS Team
 * @date    16.09.2015 (date of creation)
 * @ingroup DRIVERS_CONF
 * @prefix  CANS
 *
 * @brief   Configuration of the messages and signal settings for the CAN driver
 *
 */

```

```c
/*================== Includes ====================================================*/
#include "cansignal_cfg.h"

#include "bal.h"
#include "database.h"
#include "diag.h"
#include "sox.h"
#include "sys.h"
#include "bms.h"
#include "contactor.h"

/*================== Macros and Definitions ======================================*/
static DATA_BLOCK_CURRENT_SENSOR_s cans_current_tab;    Move to Line 106.

#define CANS_MODULSIGNALS_VOLT      (CAN0_SIG_Mod0_temp_valid_0_2 - CAN0_SIG_Mod0_volt_valid_0_2)
#define CANS_MODULSIGNALS_TEMP      (CAN0_SIG_Mod1_volt_valid_0_2 - CAN0_SIG_Mod0_temp_valid_0_2)


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

static float cans_checkLimits(float value, uint32_t sigIdx);

/* TX/Getter functions */
static uint32_t cans_getvolt(uint32_t, void *);
static uint32_t cans_gettempering(uint32_t, void *);
static uint32_t cans_getcanerr(uint32_t, void *);
static uint32_t cans_gettemp(uint32_t, void *);
static uint32_t cans_getsoc(uint32_t, void *);
static uint32_t cans_getRecommendedOperatingCurrent(uint32_t, void *);
static uint32_t cans_getMaxAllowedPower(uint32_t, void *);
static uint32_t cans_getpower(uint32_t, void *);
static uint32_t cans_getcurr(uint32_t, void *);
static uint32_t cans_getPackVoltage(uint32_t, void *);
static uint32_t cans_getminmaxvolt(uint32_t, void *);
static uint32_t cans_getminmaxtemp(uint32_t, void *);
static uint32_t cans_getisoguard(uint32_t, void *);


/* RX/Setter functions */
static uint32_t cans_setcurr(uint32_t, void *);
static uint32_t cans_setstaterequest(uint32_t, void *);
static uint32_t cans_setdebug(uint32_t, void *);
static uint32_t cans_setSWversion(uint32_t, void *);
static uint32_t cans_setenginerequest(uint32_t, void *);


#ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
static uint32_t cans_gettriggercurrent(uint32_t sigIdx, void *value);
#endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
```

```c
typedef struct  {
    CANS_messages_t msgIdx;
    uint8_t bit_position;
    uint8_t bit_length;
    float min;
    float max;
    float factor;
    float offset;
    CANS_byteOrder_e byteOrder;
    can_callback_funcPtr callback;
} CANS_signal_s;
```

```
105  /*================== Static Constant and Variable Definitions ==============*/
106
107  /*================== Extern Constant and Variable Definitions ==============*/
108                          When using CAN0_MSGIdx_xx, it is much easier to understand.
109  const CANS_signal_s cans_CAN0_signals_tx[] = {
110      { {CAN0_MSG_SystemState_0}, 0, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!< CAN0_SIG_GS0_general_error, */
111      { {CAN0_MSG_SystemState_0}, 8, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_current_state, */  Used for sending the BMS state!!!
112      { {CAN0_MSG_SystemState_0}, 16, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_overtemp_charge, */
113      { {CAN0_MSG_SystemState_0}, 24, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_undertemp_charge, */
114      { {CAN0_MSG_SystemState_0}, 32, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_overtemp_discharge, */
115      { {CAN0_MSG_SystemState_0}, 40, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_undertemp_discharge, */
116      { {CAN0_MSG_SystemState_0}, 48, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_overcurrent_charge, */
117      { {CAN0_MSG_SystemState_0}, 56, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS0_error_overcurrent_discharge, */
118
119      { {CAN0_MSG_SystemState_1}, 0, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_overvoltage, */
120      { {CAN0_MSG_SystemState_1}, 8, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_undervoltage, */
121      { {CAN0_MSG_SystemState_1}, 11, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_deep_discharge, */
122      { {CAN0_MSG_SystemState_1}, 16, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_temperature_MCU0 */
123      { {CAN0_MSG_SystemState_1}, 24, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_contactor */
124      { {CAN0_MSG_SystemState_1}, 32, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_error_selftest, */
125      { {CAN0_MSG_SystemState_1}, 40, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN_SIG_GS1_error_cantiming, */
126      { {CAN0_MSG_SystemState_1}, 48, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_current_sensor, */
127      { {CAN0_MSG_SystemState_1}, 56, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS1_balancing_active, */
128
129      { {CAN0_MSG_SystemState_2}, 0, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getcanerr },  /*!<
         CAN0_SIG_GS2_states_relays */    This is for relay open/close information?
130      { {CAN0_MSG_SystemState_2}, 16, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },   /*!<
         CAN0_SIG_GS2_error_insulation */
131      { {CAN0_MSG_SystemState_2}, 24, 4, 0, 15, 1, 0, littleEndian, &cans_getcanerr },  /*!< CAN0_SIG_GS2_fuse_state */
132      { {CAN0_MSG_SystemState_2}, 32, 2, 0, 3, 1, 0, littleEndian, &cans_getcanerr },   /*!<
         CAN0_SIG_GS2_lowCoinCellVolt */
133      { {CAN0_MSG_SystemState_2}, 40, 1, 0, 1, 1, 0, littleEndian, &cans_getcanerr },   /*!<
         CAN0_SIG_GS2_error_openWire */
134      { {CAN0_MSG_SystemState_2}, 48, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },   /*!< CAN0_SIG_GS2_daisyChain */
135      { {CAN0_MSG_SystemState_2}, 56, 3, 0, 7, 1, 0, littleEndian, &cans_getcanerr },   /*!<
         CAN0_SIG_GS2_plausibilityCheck */
```

```
136
137         { {CAN0_MSG_SlaveState_0}, 0, 64, 0, UINT64_MAX, 1, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SS0_states */
138         { {CAN0_MSG_SlaveState_1}, 0, 64, 0, UINT64_MAX, 1, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SS0_states */
139
140         { {CAN0_MSG_RecOperatingCurrent}, 0, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getRecommendedOperatingCurrent },
        /*!< CAN0_SIG_MaxChargeCurrent */
141         { {CAN0_MSG_RecOperatingCurrent}, 16, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getRecommendedOperatingCurrent
        },  /*!< CAN0_SIG_MaxChargeCurrent_Peak */
142         { {CAN0_MSG_RecOperatingCurrent}, 32, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getRecommendedOperatingCurrent
        },  /*!< CAN0_SIG_MaxDischargeCurrent */
143         { {CAN0_MSG_RecOperatingCurrent}, 48, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getRecommendedOperatingCurrent
        },  /*!< CAN0_SIG_MaxDischargeCurrent_Peak */
144
145         { {CAN0_MSG_SOP}, 0, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getMaxAllowedPower },  /*!<
        CAN0_SIG_MaxChargePower */
146         { {CAN0_MSG_SOP}, 16, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getMaxAllowedPower },  /*!<
        CAN0_SIG_MaxChargePower_Peak */
147         { {CAN0_MSG_SOP}, 32, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getMaxAllowedPower },  /*!<
        CAN0_SIG_MaxDischargePower */
148         { {CAN0_MSG_SOP}, 48, 16, 0, 6553.5, 10, 0, littleEndian, &cans_getMaxAllowedPower },  /*!<
        CAN0_SIG_MaxDischargePower_Peak */
149
150         { {CAN0_MSG_SOC}, 0, 16, 0, 100, 100, 0, littleEndian, &cans_getsoc },  /*!< CAN0_SIG_SOC_mean */
151         { {CAN0_MSG_SOC}, 16, 16, 0, 100, 100, 0, littleEndian, &cans_getsoc },  /*!< CAN0_SIG_SOC_min */
152         { {CAN0_MSG_SOC}, 32, 16, 0, 100, 100, 0, littleEndian, &cans_getsoc },  /*!< CAN0_SIG_SOC_max */
153
154         { {CAN0_MSG_SOH}, 0, 16, 0, 0, 100, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SOH_mean */
155         { {CAN0_MSG_SOH}, 16, 16, 0, 0, 100, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SOH_min */
156         { {CAN0_MSG_SOH}, 32, 16, 0, 0, 100, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SOH_max */
157
158         { {CAN0_MSG_SOE}, 0, 16, 0, 0, 100, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SOE */
159         { {CAN0_MSG_SOE}, 16, 32, 0, UINT32_MAX, 1, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_RemainingEnergy */
160
161         { {CAN0_MSG_MinMaxCellVolt}, 0, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getminmaxvolt },  /*!<
        CAN0_SIG_Cellvolt_mean */
162         { {CAN0_MSG_MinMaxCellVolt}, 16, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getminmaxvolt },  /*!<
        CAN0_SIG_Cellvolt_min */
163         { {CAN0_MSG_MinMaxCellVolt}, 32, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getminmaxvolt },  /*!<
        CAN0_SIG_Cellvolt_max */
164         { {CAN0_MSG_MinMaxCellVolt}, 48, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getminmaxvolt },  /*!<
        CAN0_SIG_ModNumber_min */
165         { {CAN0_MSG_MinMaxCellVolt}, 56, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getminmaxvolt },  /*!<
        CAN0_SIG_ModNumber_max */
166
167         { {CAN0_MSG_SOV}, 0, 16, 0, 100, 100, 0, littleEndian, NULL_PTR },  /*!< CAN0_SIG_SOV */
168
169         { {CAN0_MSG_MinMaxCellTemp}, 0, 16, -128, 527.35, 100, 128, littleEndian, &cans_getminmaxtemp },  /*!<
        CAN0_SIG_Cellvolt_mean */
170         { {CAN0_MSG_MinMaxCellTemp}, 16, 16, -128, 527.35, 100, 128, littleEndian, &cans_getminmaxtemp },  /*!<
        CAN0_SIG_Cellvolt_min */
171         { {CAN0_MSG_MinMaxCellTemp}, 32, 16, -128, 527.35, 100, 128, littleEndian, &cans_getminmaxtemp },  /*!<
        CAN0_SIG_Cellvolt_max */
```

```
172         { {CAN0_MSG_MinMaxCellTemp}, 48, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getminmaxtemp },  /*!<
        CAN0_SIG_ModNumber_min */
173         { {CAN0_MSG_MinMaxCellTemp}, 56, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getminmaxtemp },  /*!<
        CAN0_SIG_ModNumber_max */
174
175         { {CAN0_MSG_Tempering}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettempering },  /*!<
        CAN0_SIG_CoolingNeeded */
176         { {CAN0_MSG_Tempering}, 8, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettempering },  /*!<
        CAN0_SIG_HeatingNeeded */
177         { {CAN0_MSG_Tempering}, 16, 32, 0, UINT32_MAX, 1, 0, littleEndian, &cans_gettempering },  /*!<
        CAN0_SIG_TemperingDemand */
178
179         { {CAN0_MSG_Insulation}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getisoguard },  /*!<
        CAN0_SIG_InsulationStatus */
180         { {CAN0_MSG_Insulation}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getisoguard },  /*!<
        CAN0_SIG_InsulationValue */
181
182         { {CAN0_MSG_Power_0}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_1s */
183         { {CAN0_MSG_Power_0}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_5s */
184         { {CAN0_MSG_Power_1}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_10s */
185         { {CAN0_MSG_Power_1}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_30s */
186         { {CAN0_MSG_Power_2}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_60s */
187         { {CAN0_MSG_Power_2}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getpower },  /*!<
        CAN0_SIG_RunAverage_Power_config */
188
189         { {CAN0_MSG_Current_0}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_1s */
190         { {CAN0_MSG_Current_0}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_5s */
191         { {CAN0_MSG_Current_1}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_10s */
192         { {CAN0_MSG_Current_1}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_30s */
193         { {CAN0_MSG_Current_2}, 0, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_60s */
194         { {CAN0_MSG_Current_2}, 32, 32, -2500000, 4292467295, 1, 2500000, littleEndian, &cans_getcurr },  /*!<
        CAN0_SIG_RunAverage_Current_config */
195
196         { {CAN0_MSG_PackVoltage}, 0, 32, 0, UINT32_MAX, 1, 0, littleEndian, &cans_getPackVoltage },  /*!<
        CAN0_SIG_PackVolt_Battery */
197         { {CAN0_MSG_PackVoltage}, 32, 32, 0, UINT32_MAX, 1, 0, littleEndian, &cans_getPackVoltage },  /*!<
        CAN0_SIG_PackVolt_PowerNet */
198
199         /* Module 0 cell voltages */
200         { {CAN0_MSG_Mod0_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_0_2 */
201         { {CAN0_MSG_Mod0_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod0_volt_0 */
202     { {CAN0_MSG_Mod0_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_1 */
203     { {CAN0_MSG_Mod0_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_2 */
204     { {CAN0_MSG_Mod0_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_3_5 */
205     { {CAN0_MSG_Mod0_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_3 */
206     { {CAN0_MSG_Mod0_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_4 */
207     { {CAN0_MSG_Mod0_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_5 */
208     { {CAN0_MSG_Mod0_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_6_8 */
209     { {CAN0_MSG_Mod0_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_6 */
210     { {CAN0_MSG_Mod0_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_7 */
211     { {CAN0_MSG_Mod0_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_8 */
212     { {CAN0_MSG_Mod0_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_9_11 */
213     { {CAN0_MSG_Mod0_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_9 */
214     { {CAN0_MSG_Mod0_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_10 */
215     { {CAN0_MSG_Mod0_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_11 */
216     { {CAN0_MSG_Mod0_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_12_14 */
217     { {CAN0_MSG_Mod0_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_12 */
218     { {CAN0_MSG_Mod0_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_13 */
219     { {CAN0_MSG_Mod0_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_14 */
220     { {CAN0_MSG_Mod0_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_valid_15_17 */
221     { {CAN0_MSG_Mod0_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_15 */
222     { {CAN0_MSG_Mod0_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_16 */
223     { {CAN0_MSG_Mod0_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod0_volt_17 */
224
225     /* Module 0 cell temperatures */
226     { {CAN0_MSG_Mod0_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_volt_valid_0_2 */
227     { {CAN0_MSG_Mod0_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_0 */
228     { {CAN0_MSG_Mod0_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
```

```
        CAN0_SIG_Mod0_temp_1 */
229     { {CAN0_MSG_Mod0_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_2 */
230     { {CAN0_MSG_Mod0_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_volt_valid_3_5 */
231     { {CAN0_MSG_Mod0_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_3 */
232     { {CAN0_MSG_Mod0_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_4 */
233     { {CAN0_MSG_Mod0_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_5 */
234     { {CAN0_MSG_Mod0_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_volt_valid_6_8 */
235     { {CAN0_MSG_Mod0_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_6 */
236     { {CAN0_MSG_Mod0_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_7 */
237     { {CAN0_MSG_Mod0_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_8 */
238     { {CAN0_MSG_Mod0_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_volt_valid_9_11 */
239     { {CAN0_MSG_Mod0_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_9 */
240     { {CAN0_MSG_Mod0_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_10 */
241     { {CAN0_MSG_Mod0_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod0_temp_11 */
242
243     /* Module 1 cell voltages */
244     { {CAN0_MSG_Mod1_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_0_2 */
245     { {CAN0_MSG_Mod1_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_0 */
246     { {CAN0_MSG_Mod1_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_1 */
247     { {CAN0_MSG_Mod1_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_2 */
248     { {CAN0_MSG_Mod1_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_3_5 */
249     { {CAN0_MSG_Mod1_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_3 */
250     { {CAN0_MSG_Mod1_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_4 */
251     { {CAN0_MSG_Mod1_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_5 */
252     { {CAN0_MSG_Mod1_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_6_8 */
253     { {CAN0_MSG_Mod1_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_6 */
254     { {CAN0_MSG_Mod1_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_7 */
255     { {CAN0_MSG_Mod1_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod1_volt_8 */
256     { {CAN0_MSG_Mod1_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_9_11 */
257     { {CAN0_MSG_Mod1_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_9 */
258     { {CAN0_MSG_Mod1_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_10 */
259     { {CAN0_MSG_Mod1_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_11 */
260     { {CAN0_MSG_Mod1_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_12_14 */
261     { {CAN0_MSG_Mod1_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_12 */
262     { {CAN0_MSG_Mod1_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_13 */
263     { {CAN0_MSG_Mod1_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_14 */
264     { {CAN0_MSG_Mod1_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_valid_15_17 */
265     { {CAN0_MSG_Mod1_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_15 */
266     { {CAN0_MSG_Mod1_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_16 */
267     { {CAN0_MSG_Mod1_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod1_volt_17 */
268
269     /* Module 1 cell temperatures */
270     { {CAN0_MSG_Mod1_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_volt_valid_0_2 */
271     { {CAN0_MSG_Mod1_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_0 */
272     { {CAN0_MSG_Mod1_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_1 */
273     { {CAN0_MSG_Mod1_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_2 */
274     { {CAN0_MSG_Mod1_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_volt_valid_3_5 */
275     { {CAN0_MSG_Mod1_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_3 */
276     { {CAN0_MSG_Mod1_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_4 */
277     { {CAN0_MSG_Mod1_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_5 */
278     { {CAN0_MSG_Mod1_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_volt_valid_6_8 */
279     { {CAN0_MSG_Mod1_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_6 */
280     { {CAN0_MSG_Mod1_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_7 */
281     { {CAN0_MSG_Mod1_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_8 */
282     { {CAN0_MSG_Mod1_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
```

```
        CAN0_SIG_Mod1_volt_valid_9_11 */
283     { {CAN0_MSG_Mod1_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_9 */
284     { {CAN0_MSG_Mod1_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_10 */
285     { {CAN0_MSG_Mod1_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod1_temp_11 */
286
287     /* Module 2 cell voltages */
288     { {CAN0_MSG_Mod2_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_0_2 */
289     { {CAN0_MSG_Mod2_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_0 */
290     { {CAN0_MSG_Mod2_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_1 */
291     { {CAN0_MSG_Mod2_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_2 */
292     { {CAN0_MSG_Mod2_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_3_5 */
293     { {CAN0_MSG_Mod2_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_3 */
294     { {CAN0_MSG_Mod2_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_4 */
295     { {CAN0_MSG_Mod2_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_5 */
296     { {CAN0_MSG_Mod2_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_6_8 */
297     { {CAN0_MSG_Mod2_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_6 */
298     { {CAN0_MSG_Mod2_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_7 */
299     { {CAN0_MSG_Mod2_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_8 */
300     { {CAN0_MSG_Mod2_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_9_11 */
301     { {CAN0_MSG_Mod2_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_9 */
302     { {CAN0_MSG_Mod2_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_10 */
303     { {CAN0_MSG_Mod2_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_11 */
304     { {CAN0_MSG_Mod2_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_12_14 */
305     { {CAN0_MSG_Mod2_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_12 */
306     { {CAN0_MSG_Mod2_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_13 */
307     { {CAN0_MSG_Mod2_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_14 */
308     { {CAN0_MSG_Mod2_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_valid_15_17 */
309     { {CAN0_MSG_Mod2_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod2_volt_15 */
310     { {CAN0_MSG_Mod2_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_16 */
311     { {CAN0_MSG_Mod2_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod2_volt_17 */
312
313     /* Module 2 cell temperatures */
314     { {CAN0_MSG_Mod2_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_volt_valid_0_2 */
315     { {CAN0_MSG_Mod2_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_0 */
316     { {CAN0_MSG_Mod2_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_1 */
317     { {CAN0_MSG_Mod2_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_2 */
318     { {CAN0_MSG_Mod2_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_volt_valid_3_5 */
319     { {CAN0_MSG_Mod2_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_3 */
320     { {CAN0_MSG_Mod2_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_4 */
321     { {CAN0_MSG_Mod2_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_5 */
322     { {CAN0_MSG_Mod2_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_volt_valid_6_8 */
323     { {CAN0_MSG_Mod2_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_6 */
324     { {CAN0_MSG_Mod2_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_7 */
325     { {CAN0_MSG_Mod2_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_8 */
326     { {CAN0_MSG_Mod2_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_volt_valid_9_11 */
327     { {CAN0_MSG_Mod2_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_9 */
328     { {CAN0_MSG_Mod2_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_10 */
329     { {CAN0_MSG_Mod2_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod2_temp_11 */
330
331     /* Module 3 cell voltages */
332     { {CAN0_MSG_Mod3_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_0_2 */
333     { {CAN0_MSG_Mod3_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_0 */
334     { {CAN0_MSG_Mod3_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_1 */
335     { {CAN0_MSG_Mod3_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_2 */
336     { {CAN0_MSG_Mod3_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_3_5 */
337     { {CAN0_MSG_Mod3_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod3_volt_3 */
338     { {CAN0_MSG_Mod3_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_4 */
339     { {CAN0_MSG_Mod3_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_5 */
340     { {CAN0_MSG_Mod3_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_6_8 */
341     { {CAN0_MSG_Mod3_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_6 */
342     { {CAN0_MSG_Mod3_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_7 */
343     { {CAN0_MSG_Mod3_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_8 */
344     { {CAN0_MSG_Mod3_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_9_11 */
345     { {CAN0_MSG_Mod3_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_9 */
346     { {CAN0_MSG_Mod3_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_10 */
347     { {CAN0_MSG_Mod3_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_11 */
348     { {CAN0_MSG_Mod3_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_12_14 */
349     { {CAN0_MSG_Mod3_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_12 */
350     { {CAN0_MSG_Mod3_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_13 */
351     { {CAN0_MSG_Mod3_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_14 */
352     { {CAN0_MSG_Mod3_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_valid_15_17 */
353     { {CAN0_MSG_Mod3_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_15 */
354     { {CAN0_MSG_Mod3_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_16 */
355     { {CAN0_MSG_Mod3_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod3_volt_17 */
356
357     /* Module 3 cell temperatures */
358     { {CAN0_MSG_Mod3_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_volt_valid_0_2 */
359     { {CAN0_MSG_Mod3_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_0 */
360     { {CAN0_MSG_Mod3_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_1 */
361     { {CAN0_MSG_Mod3_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_2 */
362     { {CAN0_MSG_Mod3_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_volt_valid_3_5 */
363     { {CAN0_MSG_Mod3_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_3 */
364     { {CAN0_MSG_Mod3_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
```

```
        CAN0_SIG_Mod3_temp_4 */
365     { {CAN0_MSG_Mod3_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_5 */
366     { {CAN0_MSG_Mod3_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_volt_valid_6_8 */
367     { {CAN0_MSG_Mod3_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_6 */
368     { {CAN0_MSG_Mod3_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_7 */
369     { {CAN0_MSG_Mod3_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_8 */
370     { {CAN0_MSG_Mod3_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_volt_valid_9_11 */
371     { {CAN0_MSG_Mod3_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_9 */
372     { {CAN0_MSG_Mod3_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_10 */
373     { {CAN0_MSG_Mod3_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod3_temp_11 */
374
375     /* Module 4 cell voltages */
376     { {CAN0_MSG_Mod4_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_valid_0_2 */
377     { {CAN0_MSG_Mod4_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_0 */
378     { {CAN0_MSG_Mod4_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_1 */
379     { {CAN0_MSG_Mod4_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_2 */
380     { {CAN0_MSG_Mod4_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_valid_3_5 */
381     { {CAN0_MSG_Mod4_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_3 */
382     { {CAN0_MSG_Mod4_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_4 */
383     { {CAN0_MSG_Mod4_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_5 */
384     { {CAN0_MSG_Mod4_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_valid_6_8 */
385     { {CAN0_MSG_Mod4_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_6 */
386     { {CAN0_MSG_Mod4_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_7 */
387     { {CAN0_MSG_Mod4_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_8 */
388     { {CAN0_MSG_Mod4_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_valid_9_11 */
389     { {CAN0_MSG_Mod4_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_9 */
390     { {CAN0_MSG_Mod4_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod4_volt_10 */
391     { {CAN0_MSG_Mod4_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod4_volt_11 */
392     { {CAN0_MSG_Mod4_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_valid_12_14 */
393     { {CAN0_MSG_Mod4_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_12 */
394     { {CAN0_MSG_Mod4_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_13 */
395     { {CAN0_MSG_Mod4_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_14 */
396     { {CAN0_MSG_Mod4_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_valid_15_17 */
397     { {CAN0_MSG_Mod4_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_15 */
398     { {CAN0_MSG_Mod4_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_16 */
399     { {CAN0_MSG_Mod4_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod5_volt_17 */
400
401     /* Module 4 cell temperatures */
402     { {CAN0_MSG_Mod4_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_0_2 */
403     { {CAN0_MSG_Mod4_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_0 */
404     { {CAN0_MSG_Mod4_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_1 */
405     { {CAN0_MSG_Mod4_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_2 */
406     { {CAN0_MSG_Mod4_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_3_5 */
407     { {CAN0_MSG_Mod4_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_3 */
408     { {CAN0_MSG_Mod4_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_4 */
409     { {CAN0_MSG_Mod4_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_5 */
410     { {CAN0_MSG_Mod4_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_6_8 */
411     { {CAN0_MSG_Mod4_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_6 */
412     { {CAN0_MSG_Mod4_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_7 */
413     { {CAN0_MSG_Mod4_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_8 */
414     { {CAN0_MSG_Mod4_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_9_11 */
415     { {CAN0_MSG_Mod4_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_9 */
416     { {CAN0_MSG_Mod4_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_10 */
417     { {CAN0_MSG_Mod4_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_11 */
418
```

```
419         /* Module 5 cell voltages */
420         { {CAN0_MSG_Mod5_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_0_2 */
421         { {CAN0_MSG_Mod5_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_0 */
422         { {CAN0_MSG_Mod5_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_1 */
423         { {CAN0_MSG_Mod5_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_2 */
424         { {CAN0_MSG_Mod5_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_3_5 */
425         { {CAN0_MSG_Mod5_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_3 */
426         { {CAN0_MSG_Mod5_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_4 */
427         { {CAN0_MSG_Mod5_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_5 */
428         { {CAN0_MSG_Mod5_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_6_8 */
429         { {CAN0_MSG_Mod5_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_6 */
430         { {CAN0_MSG_Mod5_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_7 */
431         { {CAN0_MSG_Mod5_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_8 */
432         { {CAN0_MSG_Mod5_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_9_11 */
433         { {CAN0_MSG_Mod5_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_9 */
434         { {CAN0_MSG_Mod5_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_10 */
435         { {CAN0_MSG_Mod5_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_11 */
436         { {CAN0_MSG_Mod5_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_12_14 */
437         { {CAN0_MSG_Mod5_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_12 */
438         { {CAN0_MSG_Mod5_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_13 */
439         { {CAN0_MSG_Mod5_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_14 */
440         { {CAN0_MSG_Mod5_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_valid_15_17 */
441         { {CAN0_MSG_Mod5_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_15 */
442         { {CAN0_MSG_Mod5_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_16 */
443         { {CAN0_MSG_Mod5_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
            CAN0_SIG_Mod5_volt_17 */
444
445         /* Module 5 cell temperatures */
446         { {CAN0_MSG_Mod5_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
```

```
        CAN0_SIG_Mod5_volt_valid_0_2 */
447     { {CAN0_MSG_Mod5_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_0 */
448     { {CAN0_MSG_Mod5_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_1 */
449     { {CAN0_MSG_Mod5_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_2 */
450     { {CAN0_MSG_Mod5_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_volt_valid_3_5 */
451     { {CAN0_MSG_Mod5_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_3 */
452     { {CAN0_MSG_Mod5_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_4 */
453     { {CAN0_MSG_Mod5_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_5 */
454     { {CAN0_MSG_Mod5_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_volt_valid_6_8 */
455     { {CAN0_MSG_Mod5_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_6 */
456     { {CAN0_MSG_Mod5_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_7 */
457     { {CAN0_MSG_Mod5_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_8 */
458     { {CAN0_MSG_Mod5_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_volt_valid_9_11 */
459     { {CAN0_MSG_Mod5_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_9 */
460     { {CAN0_MSG_Mod5_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_10 */
461     { {CAN0_MSG_Mod5_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod5_temp_11 */
462
463     /* Module 6 cell voltages */
464     { {CAN0_MSG_Mod6_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_0_2 */
465     { {CAN0_MSG_Mod6_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_0 */
466     { {CAN0_MSG_Mod6_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_1 */
467     { {CAN0_MSG_Mod6_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_2 */
468     { {CAN0_MSG_Mod6_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_3_5 */
469     { {CAN0_MSG_Mod6_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_3 */
470     { {CAN0_MSG_Mod6_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_4 */
471     { {CAN0_MSG_Mod6_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_5 */
472     { {CAN0_MSG_Mod6_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_6_8 */
473     { {CAN0_MSG_Mod6_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod6_volt_6 */
474     { {CAN0_MSG_Mod6_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_7 */
475     { {CAN0_MSG_Mod6_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_8 */
476     { {CAN0_MSG_Mod6_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_9_11 */
477     { {CAN0_MSG_Mod6_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_9 */
478     { {CAN0_MSG_Mod6_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_10 */
479     { {CAN0_MSG_Mod6_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_11 */
480     { {CAN0_MSG_Mod6_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_12_14 */
481     { {CAN0_MSG_Mod6_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_12 */
482     { {CAN0_MSG_Mod6_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_13 */
483     { {CAN0_MSG_Mod6_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_14 */
484     { {CAN0_MSG_Mod6_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_valid_15_17 */
485     { {CAN0_MSG_Mod6_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_15 */
486     { {CAN0_MSG_Mod6_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_16 */
487     { {CAN0_MSG_Mod6_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod6_volt_17 */
488
489     /* Module 6 cell temperatures */
490     { {CAN0_MSG_Mod6_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_volt_valid_0_2 */
491     { {CAN0_MSG_Mod6_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_0 */
492     { {CAN0_MSG_Mod6_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_1 */
493     { {CAN0_MSG_Mod6_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_2 */
494     { {CAN0_MSG_Mod6_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_volt_valid_3_5 */
495     { {CAN0_MSG_Mod6_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_3 */
496     { {CAN0_MSG_Mod6_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_4 */
497     { {CAN0_MSG_Mod6_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_5 */
498     { {CAN0_MSG_Mod6_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_volt_valid_6_8 */
499     { {CAN0_MSG_Mod6_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_6 */
500     { {CAN0_MSG_Mod6_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
```

```
        CAN0_SIG_Mod6_temp_7 */
501     { {CAN0_MSG_Mod6_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_8 */
502     { {CAN0_MSG_Mod6_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_volt_valid_9_11 */
503     { {CAN0_MSG_Mod6_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_9 */
504     { {CAN0_MSG_Mod6_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_10 */
505     { {CAN0_MSG_Mod6_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod6_temp_11 */
506
507     /* Module 7 cell voltages */
508     { {CAN0_MSG_Mod7_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_0_2 */
509     { {CAN0_MSG_Mod7_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_0 */
510     { {CAN0_MSG_Mod7_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_1 */
511     { {CAN0_MSG_Mod7_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_2 */
512     { {CAN0_MSG_Mod7_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_3_5 */
513     { {CAN0_MSG_Mod7_Cellvolt_1}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_3 */
514     { {CAN0_MSG_Mod7_Cellvolt_1}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_4 */
515     { {CAN0_MSG_Mod7_Cellvolt_1}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_5 */
516     { {CAN0_MSG_Mod7_Cellvolt_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_6_8 */
517     { {CAN0_MSG_Mod7_Cellvolt_2}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_6 */
518     { {CAN0_MSG_Mod7_Cellvolt_2}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_7 */
519     { {CAN0_MSG_Mod7_Cellvolt_2}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_8 */
520     { {CAN0_MSG_Mod7_Cellvolt_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_9_11 */
521     { {CAN0_MSG_Mod7_Cellvolt_3}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_9 */
522     { {CAN0_MSG_Mod7_Cellvolt_3}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_10 */
523     { {CAN0_MSG_Mod7_Cellvolt_3}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_11 */
524     { {CAN0_MSG_Mod7_Cellvolt_4}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_12_14 */
525     { {CAN0_MSG_Mod7_Cellvolt_4}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_12 */
526     { {CAN0_MSG_Mod7_Cellvolt_4}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_13 */
527     { {CAN0_MSG_Mod7_Cellvolt_4}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
```

```
        CAN0_SIG_Mod7_volt_14 */
528     { {CAN0_MSG_Mod7_Cellvolt_5}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_valid_15_17 */
529     { {CAN0_MSG_Mod7_Cellvolt_5}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_15 */
530     { {CAN0_MSG_Mod7_Cellvolt_5}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_16 */
531     { {CAN0_MSG_Mod7_Cellvolt_5}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },  /*!<
        CAN0_SIG_Mod7_volt_17 */
532
533     /* Module 7 cell temperatures */
534     { {CAN0_MSG_Mod7_Celltemp_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_0_2 */
535     { {CAN0_MSG_Mod7_Celltemp_0}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_0 */
536     { {CAN0_MSG_Mod7_Celltemp_0}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_1 */
537     { {CAN0_MSG_Mod7_Celltemp_0}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_2 */
538     { {CAN0_MSG_Mod7_Celltemp_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_3_5 */
539     { {CAN0_MSG_Mod7_Celltemp_1}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_3 */
540     { {CAN0_MSG_Mod7_Celltemp_1}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_4 */
541     { {CAN0_MSG_Mod7_Celltemp_1}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_5 */
542     { {CAN0_MSG_Mod7_Celltemp_2}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_6_8 */
543     { {CAN0_MSG_Mod7_Celltemp_2}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_6 */
544     { {CAN0_MSG_Mod7_Celltemp_2}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_7 */
545     { {CAN0_MSG_Mod7_Celltemp_2}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_8 */
546     { {CAN0_MSG_Mod7_Celltemp_3}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_volt_valid_9_11 */
547     { {CAN0_MSG_Mod7_Celltemp_3}, 8, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_9 */
548     { {CAN0_MSG_Mod7_Celltemp_3}, 24, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_10 */
549     { {CAN0_MSG_Mod7_Celltemp_3}, 40, 16, -128, 527.35, 100, 128, littleEndian, &cans_gettemp },  /*!<
        CAN0_SIG_Mod4_temp_11 */
550
551 #ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
552         {{CAN0_MSG_BMS_CurrentTrigger}, 0, 32, 0, 0, 1, 0, &cans_gettriggercurrent }  /*!< CAN0_SIG_ISA_Trigger */
553 #endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
554 };
555
556
557 const CANS_signal_s cans_CAN1_signals_tx[] = {
558 };
```

Indexes for the CAN signals:

cansignal_cfg.h × | can_cfg.c | can.h | can.c | stm

mcu-primary > src > module > config > cansignal_cfg.h > CANS_signal_s

```
241  typedef enum {
242      CAN0_SIG_GS0_general_error,  /* 0:good, 1:error */
243      CAN0_SIG_GS0_current_state,  /* currently no used */
244      CAN0_SIG_GS0_error_overtemp_charge,  /* 0:good, 1:error */
```

cansignal.c | cansignal_cfg.h × | cansign

mcu-primary > src > module > config > cansignal_cfg

```
333      CAN0_SIG_Mod0_volt_valid_0_2,
334      CAN0_SIG_Mod0_volt_0,
335      CAN0_SIG_Mod0_volt_1,
336      CAN0_SIG_Mod0_volt_2,
337      CAN0_SIG_Mod0_volt_valid_3_5,
338      CAN0_SIG_Mod0_volt_3,
339      CAN0_SIG_Mod0_volt_4,
340      CAN0_SIG_Mod0_volt_5,
341      CAN0_SIG_Mod0_volt_valid_6_8,
342      CAN0_SIG_Mod0_volt_6,
343      CAN0_SIG_Mod0_volt_7,
344      CAN0_SIG_Mod0_volt_8,
345      CAN0_SIG_Mod0_volt_valid_9_11,
346      CAN0_SIG_Mod0_volt_9,
347      CAN0_SIG_Mod0_volt_10,
348      CAN0_SIG_Mod0_volt_11,
349      CAN0_SIG_Mod0_volt_valid_12_14,
350      CAN0_SIG_Mod0_volt_12,
351      CAN0_SIG_Mod0_volt_13,
352      CAN0_SIG_Mod0_volt_14,
353      CAN0_SIG_Mod0_volt_valid_15_17,
354      CAN0_SIG_Mod0_volt_15,
355      CAN0_SIG_Mod0_volt_16,
356      CAN0_SIG_Mod0_volt_17,
357
358      CAN0_SIG_Mod0_temp_valid_0_2,
```

cansignal_cfg.h × | can_cfg.c | can.h | can.c

mcu-primary > src > module > config > cansignal_cfg.h > CANS_signal_s

```
665      CAN0_SIG_Mod7_temp_9,
666      CAN0_SIG_Mod7_temp_10,
667      CAN0_SIG_Mod7_temp_11,
668
669  #ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
670      CAN0_SIG_ISA_Trigger,
671  #endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
672
673      CAN0_SIGNAL_NONE = 0xFFFF
674  } CANS_CAN0_signalsTx_e;
```

stm32f4xx_hal_can.h | Search: can_buffe... | cansignal_cfg.h ×

mcu-primary > src > module > config > cansignal_cfg.h > CANS_signal_s

```
754  typedef struct  {
755      CANS_messages_t msgIdx;
756      uint8_t bit_position;
757      uint8_t bit_length;
758      float min;
759      float max;
760      float factor;
761      float offset;
762      CANS_byteOrder_e byteOrder;
763      can_callback_funcPtr callback;
764  } CANS_signal_s;      You, a month ago • Add all foxBM
```

```
559
560
561  const CANS_signal_s cans_CAN0_signals_rx[] = {
562      { {CAN0_MSG_StateRequest}, 8, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_setstaterequest },
563      { {CAN0_MSG_IVT_Current}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS0_I_MuxID */
564      { {CAN0_MSG_IVT_Current}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS0_I_Status */
565      { {CAN0_MSG_IVT_Current}, 16, 32, INT32_MIN, INT32_MAX, 1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS0_I_Measurement */
566      { {CAN0_MSG_IVT_Voltage_1}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS1_U1_MuxID */
567      { {CAN0_MSG_IVT_Voltage_1}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS1_U1_Status */
568      { {CAN0_MSG_IVT_Voltage_1}, 16, 32, 0, INT32_MAX, 1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS1_U1_Measurement */
569      { {CAN0_MSG_IVT_Voltage_2}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS2_U2_MuxID */
570      { {CAN0_MSG_IVT_Voltage_2}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS2_U2_Status */
571      { {CAN0_MSG_IVT_Voltage_2}, 16, 32, 0, INT32_MAX, 1, 0, bigEndian, &cans_setcurr, },  /*
         CAN0_SIG_ISENS2_U2_Measurement */
572      { {CAN0_MSG_IVT_Voltage_3}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS3_U3_MuxID */
573      { {CAN0_MSG_IVT_Voltage_3}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS3_U3_Status */
574      { {CAN0_MSG_IVT_Voltage_3}, 16, 32, 0, INT32_MAX, 1, 0, bigEndian, &cans_setcurr, },  /*
         CAN0_SIG_ISENS3_U3_Measurement */
575      { {CAN0_MSG_IVT_Temperature}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS4_T_MuxID */
576      { {CAN0_MSG_IVT_Temperature}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS4_T_Status */
577      { {CAN0_MSG_IVT_Temperature}, 16, 32, INT32_MIN, INT32_MAX, 0.1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS4_T_Measurement */
578      { {CAN0_MSG_IVT_Power}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS5_P_MuxID */
579      { {CAN0_MSG_IVT_Power}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS5_P_Status */
580      { {CAN0_MSG_IVT_Power}, 16, 32, INT32_MIN, INT32_MAX, 1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS5_P_Measurement */
581      { {CAN0_MSG_IVT_CoulombCount}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS6_CC_MuxID */
582      { {CAN0_MSG_IVT_CoulombCount}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS6_CC_Status */
583      { {CAN0_MSG_IVT_CoulombCount}, 16, 32, INT32_MIN, INT32_MAX, 1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS6_CC_Measurement */
584      { {CAN0_MSG_IVT_EnergyCount}, 0, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS7_EC_MuxID */
585      { {CAN0_MSG_IVT_EnergyCount}, 8, 8, 0, UINT8_MAX, 1, 0, bigEndian, NULL_PTR },  /* CAN0_SIG_ISENS7_EC_Status */
586      { {CAN0_MSG_IVT_EnergyCount}, 16, 32, INT32_MIN, INT32_MAX, 1, 0, bigEndian, &cans_setcurr },  /*
         CAN0_SIG_ISENS7_EC_Measurement */
587      { {CAN0_MSG_DEBUG}, 0, 64, 0, UINT64_MAX, 1, 0, littleEndian, &cans_setdebug },  /* CAN0_SIG_DEBUG_Data */
588      { {CAN0_MSG_GetReleaseVersion}, 0, 64, 0, UINT64_MAX, 1, 0, littleEndian, &cans_setSWversion },  /*
         CAN0_SIG_DEBUG_Data */
589      { {CAN0_MSG_EngineRequest}, 0, 8, 0, UINT8_MAX, 1, 0, littleEnd
590  };
591
592  const CANS_signal_s cans_CAN1_signals_rx[] = {
593  };
594
595
596  const uint16_t cans_CAN0_signals_tx_length = sizeof(cans_CAN0_signals_tx)/sizeof(cans_CAN0_signals_tx[0]);
597  const uint16_t cans_CAN1_signals_tx_length = sizeof(cans_CAN1_signals_tx)/sizeof(cans_CAN1_signals_tx[0]);
598
599  const uint16_t cans_CAN0_signals_rx_length = sizeof(cans_CAN0_signals_rx)/sizeof(cans_CAN0_signals_rx[0]);
600  const uint16_t cans_CAN1_signals_rx_length = sizeof(cans_CAN1_signals_rx)/sizeof(cans_CAN1_signals_rx[0]);
601
```

cansignal.c × | cansignal_cfg.h | can_cfg.c | can.h | can.c

mcu-common > src > module > cansignal > cansignal.c > CANS_ParseMessage(CAN_Node

```
342             /* simple, not multiplexed signal */
343             uint64_t value = 0;
344             if (cans_signals_tx[i].callback != NULL_PTR) {
345                 cans_signals_tx[i].callback(i, &value);
346             }
347             CANS_SetSignalData(cans_signals_tx[i], value, dataptr);
```

```
602  /*================== Static Function Implementations ========================*/
603                                  Called in Line 345
604  static uint32_t cans_getvolt(uint32_t sigIdx, void *value) {
605      static DATA_BLOCK_CELLVOLTAGE_s volt_tab;
606      uint16_t modIdx = 0;
607      uint32_t cellIdx = 0;
608      uint32_t tmp = 0;
609      uint32_t tmpVal = 0;
610      float canData = 0;        No need to have a floating point number.
611
612      /* first signal to transmit cell voltages */
613      if (sigIdx == CAN0_SIG_Mod0_volt_valid_0_2) {
614          DB_ReadBlock(&volt_tab, DATA_BLOCK_ID_CELLVOLTAGE);
615      }
616
617      /* Determine module and cell number */
618      if (sigIdx - CAN0_SIG_Mod0_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
619          modIdx = 0;
620          cellIdx = sigIdx - CAN0_SIG_Mod0_volt_valid_0_2;
621      } else if (sigIdx - CAN0_SIG_Mod1_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
622          modIdx = 1;
623          cellIdx = sigIdx - CAN0_SIG_Mod1_volt_valid_0_2;
624      } else if (sigIdx - CAN0_SIG_Mod2_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
625          modIdx = 2;
626          cellIdx = sigIdx - CAN0_SIG_Mod2_volt_valid_0_2;
627      } else if (sigIdx - CAN0_SIG_Mod3_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
628          modIdx = 3;
629          cellIdx = sigIdx - CAN0_SIG_Mod3_volt_valid_0_2;
630      } else if (sigIdx - CAN0_SIG_Mod4_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
631          modIdx = 4;
632          cellIdx = sigIdx - CAN0_SIG_Mod4_volt_valid_0_2;
633      } else if (sigIdx - CAN0_SIG_Mod5_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
634          modIdx = 5;
635          cellIdx = sigIdx - CAN0_SIG_Mod5_volt_valid_0_2;
636      } else if (sigIdx - CAN0_SIG_Mod6_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
637          modIdx = 6;
638          cellIdx = sigIdx - CAN0_SIG_Mod6_volt_valid_0_2;
639      } else if (sigIdx - CAN0_SIG_Mod7_volt_valid_0_2 < CANS_MODULSIGNALS_VOLT) {
640          modIdx = 7;
641          cellIdx = sigIdx - CAN0_SIG_Mod7_volt_valid_0_2;
642      }
643
644      if (value != NULL_PTR) {
645          switch (sigIdx) {
646              case CAN0_SIG_Mod0_volt_valid_0_2:
647              case CAN0_SIG_Mod1_volt_valid_0_2:
648              case CAN0_SIG_Mod2_volt_valid_0_2:
649              case CAN0_SIG_Mod3_volt_valid_0_2:
650              case CAN0_SIG_Mod4_volt_valid_0_2:  /* Valid flags for cell voltages 0 - 2 */
651              case CAN0_SIG_Mod5_volt_valid_0_2:
652              case CAN0_SIG_Mod6_volt_valid_0_2:
653              case CAN0_SIG_Mod7_volt_valid_0_2:
```

cansignal_cfg.h | can_cfg.c | can.h

mcu-primary > src > module > config > cansignal_cfg.h

```
333      CAN0_SIG_Mod0_volt_valid_0_2,
334      CAN0_SIG_Mod0_volt_0,
335      CAN0_SIG_Mod0_volt_1,
336      CAN0_SIG_Mod0_volt_2,
337      CAN0_SIG_Mod0_volt_valid_3_5,
```

cansignal_cfg.c | cansignal.c | Search: CANS_add | cansignal.h | contactor.h | bms_cfg.h

mcu-primary > src > module > config > cansignal_cfg.c > cans_CAN0_signals_rx

```
201    /* Module 0 cell voltages */
202    { {CAN0_MSG_Mod0_Cellvolt_0}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },
203    { {CAN0_MSG_Mod0_Cellvolt_0}, 8, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },
204    { {CAN0_MSG_Mod0_Cellvolt_0}, 24, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },
205    { {CAN0_MSG_Mod0_Cellvolt_0}, 40, 16, 0, UINT16_MAX, 1, 0, littleEndian, &cans_getvolt },
206    { {CAN0_MSG_Mod0_Cellvolt_1}, 0, 8, 0, UINT8_MAX, 1, 0, littleEndian, &cans_getvolt },
```

cansignal.c | cansignal_cfg.h | cansignal_cfg.c | can_cfg.c | can.h | can.c | stm

mcu-primary > src > module > config > cansignal_cfg.c > CANS_MODULSIGNALS_VOLT

```
67   #define CANS_MODULSIGNALS_VOLT      (CAN0_SIG_Mod0_temp_valid_0_2 - CAN0_SIG_Mod0_volt_valid_0_2)
68   #define CANS_MODULSIGNALS_TEMP      (CAN0_SIG_Mod1_volt_valid_0_2 - CAN0_SIG_Mod0_temp_valid_0_2)
```

```
654                 if (modIdx >= BS_NR_OF_MODULES) {
655                     tmpVal = CAN_DEFAULT_VALID_FLAG;
656                 } else {
657                     tmp = volt_tab.valid_volt[modIdx];
658                 }
659                 *(uint32_t *)value = 0x07 & tmp;        Bug!
660                 break;
661
662             case CAN0_SIG_Mod0_volt_valid_3_5:
663             case CAN0_SIG_Mod1_volt_valid_3_5:
664             case CAN0_SIG_Mod2_volt_valid_3_5:
665             case CAN0_SIG_Mod3_volt_valid_3_5:
666             case CAN0_SIG_Mod4_volt_valid_3_5:  /* Valid flags for cell voltages 3 - 5 */
667             case CAN0_SIG_Mod5_volt_valid_3_5:
668             case CAN0_SIG_Mod6_volt_valid_3_5:
669             case CAN0_SIG_Mod7_volt_valid_3_5:
670                 if (modIdx >= BS_NR_OF_MODULES) {
671                     tmpVal = CAN_DEFAULT_VALID_FLAG;
672                 } else {
673                     tmp = volt_tab.valid_volt[modIdx] >> 3;
674                 }
675                 tmpVal = 0x07 & tmp;
676                 break;
677
678             case CAN0_SIG_Mod0_volt_valid_6_8:
679             case CAN0_SIG_Mod1_volt_valid_6_8:
680             case CAN0_SIG_Mod2_volt_valid_6_8:
681             case CAN0_SIG_Mod3_volt_valid_6_8:
682             case CAN0_SIG_Mod4_volt_valid_6_8:  /* Valid flags for cell voltages 6 - 8 */
683             case CAN0_SIG_Mod5_volt_valid_6_8:
684             case CAN0_SIG_Mod6_volt_valid_6_8:
685             case CAN0_SIG_Mod7_volt_valid_6_8:
686                 if (modIdx >= BS_NR_OF_MODULES) {
687                     tmpVal = CAN_DEFAULT_VALID_FLAG;
688                 } else {
689                     tmp = volt_tab.valid_volt[modIdx] >> 6;
690                 }
691                 tmpVal = 0x07 & tmp;
692                 break;
693
694             case CAN0_SIG_Mod0_volt_valid_9_11:
695             case CAN0_SIG_Mod1_volt_valid_9_11:
696             case CAN0_SIG_Mod2_volt_valid_9_11:
697             case CAN0_SIG_Mod3_volt_valid_9_11:
698             case CAN0_SIG_Mod4_volt_valid_9_11:  /* Valid flags for cell voltages 9 - 11 */
699             case CAN0_SIG_Mod5_volt_valid_9_11:
700             case CAN0_SIG_Mod6_volt_valid_9_11:
701             case CAN0_SIG_Mod7_volt_valid_9_11:
702                 if (modIdx >= BS_NR_OF_MODULES) {
703                     tmpVal = CAN_DEFAULT_VALID_FLAG;
704                 } else {
705                     tmp = volt_tab.valid_volt[modIdx] >> 9;
```

```
706                 }
707                 tmpVal = 0x07 & tmp;
708                 break;
709
710             case CAN0_SIG_Mod0_volt_valid_12_14:
711             case CAN0_SIG_Mod1_volt_valid_12_14:
712             case CAN0_SIG_Mod2_volt_valid_12_14:
713             case CAN0_SIG_Mod3_volt_valid_12_14:
714             case CAN0_SIG_Mod4_volt_valid_12_14:  /* Valid flags for cell voltages 12 - 14 */
715             case CAN0_SIG_Mod5_volt_valid_12_14:
716             case CAN0_SIG_Mod6_volt_valid_12_14:
717             case CAN0_SIG_Mod7_volt_valid_12_14:
718                 if (modIdx >= BS_NR_OF_MODULES) {
719                     tmpVal = CAN_DEFAULT_VALID_FLAG;
720                 } else {
721                     tmp = volt_tab.valid_volt[modIdx] >> 12;
722                 }
723                 tmpVal = 0x07 & tmp;
724                 break;
725
726             case CAN0_SIG_Mod0_volt_valid_15_17:
727             case CAN0_SIG_Mod1_volt_valid_15_17:
728             case CAN0_SIG_Mod2_volt_valid_15_17:
729             case CAN0_SIG_Mod3_volt_valid_15_17:
730             case CAN0_SIG_Mod4_volt_valid_15_17:  /* Valid flags for cell voltages 15 - 17 */
731             case CAN0_SIG_Mod5_volt_valid_15_17:
732             case CAN0_SIG_Mod6_volt_valid_15_17:
733             case CAN0_SIG_Mod7_volt_valid_15_17:
734                 if (modIdx >= BS_NR_OF_MODULES) {
735                     tmpVal = CAN_DEFAULT_VALID_FLAG;
736                 } else {
737                     tmp = volt_tab.valid_volt[modIdx] >> 15;
738                 }
739                 tmpVal = 0x07 & tmp;
740                 break;
741
742             case CAN0_SIG_Mod0_volt_0:
743             case CAN0_SIG_Mod0_volt_1:
744             case CAN0_SIG_Mod0_volt_2:
745             case CAN0_SIG_Mod1_volt_0:
746             case CAN0_SIG_Mod1_volt_1:
747             case CAN0_SIG_Mod1_volt_2:
748             case CAN0_SIG_Mod2_volt_0:
749             case CAN0_SIG_Mod2_volt_1:
750             case CAN0_SIG_Mod2_volt_2:
751             case CAN0_SIG_Mod3_volt_0:
752             case CAN0_SIG_Mod3_volt_1:
753             case CAN0_SIG_Mod3_volt_2:
754             case CAN0_SIG_Mod4_volt_0:
755             case CAN0_SIG_Mod4_volt_1:
756             case CAN0_SIG_Mod4_volt_2:
757             case CAN0_SIG_Mod5_volt_0:
```

```
758             case CAN0_SIG_Mod5_volt_1:
759             case CAN0_SIG_Mod5_volt_2:
760             case CAN0_SIG_Mod6_volt_0:
761             case CAN0_SIG_Mod6_volt_1:
762             case CAN0_SIG_Mod6_volt_2:
763             case CAN0_SIG_Mod7_volt_0:
764             case CAN0_SIG_Mod7_volt_1:
765             case CAN0_SIG_Mod7_volt_2:
766                 cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
767                 if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
768                     tmpVal = CAN_DEFAULT_VOLTAGE;
769                 } else {
770                     tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
771                 }
772                 break;
773
774             case CAN0_SIG_Mod0_volt_3:
775             case CAN0_SIG_Mod0_volt_4:
776             case CAN0_SIG_Mod0_volt_5:
777             case CAN0_SIG_Mod1_volt_3:
778             case CAN0_SIG_Mod1_volt_4:
779             case CAN0_SIG_Mod1_volt_5:
780             case CAN0_SIG_Mod2_volt_3:
781             case CAN0_SIG_Mod2_volt_4:
782             case CAN0_SIG_Mod2_volt_5:
783             case CAN0_SIG_Mod3_volt_3:
784             case CAN0_SIG_Mod3_volt_4:
785             case CAN0_SIG_Mod3_volt_5:
786             case CAN0_SIG_Mod4_volt_3:
787             case CAN0_SIG_Mod4_volt_4:
788             case CAN0_SIG_Mod4_volt_5:
789             case CAN0_SIG_Mod5_volt_3:
790             case CAN0_SIG_Mod5_volt_4:
791             case CAN0_SIG_Mod5_volt_5:
792             case CAN0_SIG_Mod6_volt_3:
793             case CAN0_SIG_Mod6_volt_4:
794             case CAN0_SIG_Mod6_volt_5:
795             case CAN0_SIG_Mod7_volt_3:
796             case CAN0_SIG_Mod7_volt_4:
797             case CAN0_SIG_Mod7_volt_5:
798                 cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
799                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_3_5 */
800                 if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
801                     tmpVal = CAN_DEFAULT_VOLTAGE;
802                 } else {
803                     tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
804                 }
805                 break;
806
807             case CAN0_SIG_Mod0_volt_6:
808             case CAN0_SIG_Mod0_volt_7:
809             case CAN0_SIG_Mod0_volt_8:
```

```
            case CAN0_SIG_Mod1_volt_6:
            case CAN0_SIG_Mod1_volt_7:
            case CAN0_SIG_Mod1_volt_8:
            case CAN0_SIG_Mod2_volt_6:
            case CAN0_SIG_Mod2_volt_7:
            case CAN0_SIG_Mod2_volt_8:
            case CAN0_SIG_Mod3_volt_6:
            case CAN0_SIG_Mod3_volt_7:
            case CAN0_SIG_Mod3_volt_8:
            case CAN0_SIG_Mod4_volt_6:
            case CAN0_SIG_Mod4_volt_7:
            case CAN0_SIG_Mod4_volt_8:
            case CAN0_SIG_Mod5_volt_6:
            case CAN0_SIG_Mod5_volt_7:
            case CAN0_SIG_Mod5_volt_8:
            case CAN0_SIG_Mod6_volt_6:
            case CAN0_SIG_Mod6_volt_7:
            case CAN0_SIG_Mod6_volt_8:
            case CAN0_SIG_Mod7_volt_6:
            case CAN0_SIG_Mod7_volt_7:
            case CAN0_SIG_Mod7_volt_8:
                cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
                cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_3_5 */
                cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_6_8 */
                if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
                    tmpVal = CAN_DEFAULT_VOLTAGE;
                } else {
                    tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
                }
                break;

            case CAN0_SIG_Mod0_volt_9:
            case CAN0_SIG_Mod0_volt_10:
            case CAN0_SIG_Mod0_volt_11:
            case CAN0_SIG_Mod1_volt_9:
            case CAN0_SIG_Mod1_volt_10:
            case CAN0_SIG_Mod1_volt_11:
            case CAN0_SIG_Mod2_volt_9:
            case CAN0_SIG_Mod2_volt_10:
            case CAN0_SIG_Mod2_volt_11:
            case CAN0_SIG_Mod3_volt_9:
            case CAN0_SIG_Mod3_volt_10:
            case CAN0_SIG_Mod3_volt_11:
            case CAN0_SIG_Mod4_volt_9:
            case CAN0_SIG_Mod4_volt_10:
            case CAN0_SIG_Mod4_volt_11:
            case CAN0_SIG_Mod5_volt_9:
            case CAN0_SIG_Mod5_volt_10:
            case CAN0_SIG_Mod5_volt_11:
            case CAN0_SIG_Mod6_volt_9:
            case CAN0_SIG_Mod6_volt_10:
            case CAN0_SIG_Mod6_volt_11:
```

```
862             case CAN0_SIG_Mod7_volt_9:
863             case CAN0_SIG_Mod7_volt_10:
864             case CAN0_SIG_Mod7_volt_11:
865                 cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
866                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_3_5 */
867                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_6_8 */
868                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_9_11 */
869                 if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
870                     tmpVal = CAN_DEFAULT_VOLTAGE;
871                 } else {
872                     tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
873                 }
874                 break;
875
876             case CAN0_SIG_Mod0_volt_12:
877             case CAN0_SIG_Mod0_volt_13:
878             case CAN0_SIG_Mod0_volt_14:
879             case CAN0_SIG_Mod1_volt_12:
880             case CAN0_SIG_Mod1_volt_13:
881             case CAN0_SIG_Mod1_volt_14:
882             case CAN0_SIG_Mod2_volt_12:
883             case CAN0_SIG_Mod2_volt_13:
884             case CAN0_SIG_Mod2_volt_14:
885             case CAN0_SIG_Mod3_volt_12:
886             case CAN0_SIG_Mod3_volt_13:
887             case CAN0_SIG_Mod3_volt_14:
888             case CAN0_SIG_Mod4_volt_12:
889             case CAN0_SIG_Mod4_volt_13:
890             case CAN0_SIG_Mod4_volt_14:
891             case CAN0_SIG_Mod5_volt_12:
892             case CAN0_SIG_Mod5_volt_13:
893             case CAN0_SIG_Mod5_volt_14:
894             case CAN0_SIG_Mod6_volt_12:
895             case CAN0_SIG_Mod6_volt_13:
896             case CAN0_SIG_Mod6_volt_14:
897             case CAN0_SIG_Mod7_volt_12:
898             case CAN0_SIG_Mod7_volt_13:
899             case CAN0_SIG_Mod7_volt_14:
900                 cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
901                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_3_5 */
902                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_6_8 */
903                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_9_11 */
904                 cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_12_14 */
905                 if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
906                     tmpVal = CAN_DEFAULT_VOLTAGE;
907                 } else {
908                     tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
909                 }
910                 break;
911
912             case CAN0_SIG_Mod0_volt_15:
913             case CAN0_SIG_Mod0_volt_16:
```

```
914                 case CAN0_SIG_Mod0_volt_17:
915                 case CAN0_SIG_Mod1_volt_15:
916                 case CAN0_SIG_Mod1_volt_16:
917                 case CAN0_SIG_Mod1_volt_17:
918                 case CAN0_SIG_Mod2_volt_15:
919                 case CAN0_SIG_Mod2_volt_16:
920                 case CAN0_SIG_Mod2_volt_17:
921                 case CAN0_SIG_Mod3_volt_15:
922                 case CAN0_SIG_Mod3_volt_16:
923                 case CAN0_SIG_Mod3_volt_17:
924                 case CAN0_SIG_Mod4_volt_15:
925                 case CAN0_SIG_Mod4_volt_16:
926                 case CAN0_SIG_Mod4_volt_17:
927                 case CAN0_SIG_Mod5_volt_15:
928                 case CAN0_SIG_Mod5_volt_16:
929                 case CAN0_SIG_Mod5_volt_17:
930                 case CAN0_SIG_Mod6_volt_15:
931                 case CAN0_SIG_Mod6_volt_16:
932                 case CAN0_SIG_Mod6_volt_17:
933                 case CAN0_SIG_Mod7_volt_15:
934                 case CAN0_SIG_Mod7_volt_16:
935                 case CAN0_SIG_Mod7_volt_17:
936                     cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
937                     cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_3_5 */
938                     cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_6_8 */
939                     cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_9_11 */
940                     cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_12_14 */
941                     cellIdx--;  /* Because of signal: CAN0_SIG_Modx_volt_valid_15_17 */
942                     if ((modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx >= BS_NR_OF_BAT_CELLS) {
943                         tmpVal = CAN_DEFAULT_VOLTAGE;
944                     } else {
945                         tmpVal = volt_tab.voltage[(modIdx * BS_NR_OF_BAT_CELLS_PER_MODULE) + cellIdx];
946                     }
947                     break;
948 
949                 default:
950                     break;
951             }
952             /* Check limits */
953             canData = cans_checkLimits((float)tmpVal, sigIdx);
954             /* Apply offset and factor */
955             *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                cans_CAN0_signals_tx[sigIdx].factor);
956         }
957 
958         return 0;
959     }
960 
961     uint32_t cans_gettemp(uint32_t sigIdx, void *value) {
962         static DATA_BLOCK_CELLTEMPERATURE_s temp_tab;
963         uint16_t modIdx = 0;
964         uint32_t cellIdx = 0;
```

```
965         uint32_t tmp = 0;
966         float tmpVal = 0;
967         float canData = 0;
968 
969         /* first signal to transmit cell temperatures */
970         if (sigIdx == CAN0_SIG_Mod0_temp_valid_0_2) {
971             DB_ReadBlock(&temp_tab, DATA_BLOCK_ID_CELLTEMPERATURE);
972         }
973 
974         /* Determine module and cell number */
975         if (sigIdx - CAN0_SIG_Mod0_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
976             modIdx = 0;
977             cellIdx = sigIdx - CAN0_SIG_Mod0_temp_valid_0_2;
978         } else if (sigIdx - CAN0_SIG_Mod1_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
979             modIdx = 1;
980             cellIdx = sigIdx - CAN0_SIG_Mod1_temp_valid_0_2;
981         } else if (sigIdx - CAN0_SIG_Mod2_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
982             modIdx = 2;
983             cellIdx = sigIdx - CAN0_SIG_Mod2_temp_valid_0_2;
984         } else if (sigIdx - CAN0_SIG_Mod3_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
985             modIdx = 3;
986             cellIdx = sigIdx - CAN0_SIG_Mod3_temp_valid_0_2;
987         } else if (sigIdx - CAN0_SIG_Mod4_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
988             modIdx = 4;
989             cellIdx = sigIdx - CAN0_SIG_Mod4_temp_valid_0_2;
990         } else if (sigIdx - CAN0_SIG_Mod5_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
991             modIdx = 5;
992             cellIdx = sigIdx - CAN0_SIG_Mod5_temp_valid_0_2;
993         } else if (sigIdx - CAN0_SIG_Mod6_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
994             modIdx = 6;
995             cellIdx = sigIdx - CAN0_SIG_Mod6_temp_valid_0_2;
996         } else if (sigIdx - CAN0_SIG_Mod7_temp_valid_0_2 < CANS_MODULSIGNALS_TEMP) {
997             modIdx = 7;
998             cellIdx = sigIdx - CAN0_SIG_Mod7_temp_valid_0_2;
999         }
1000 
1001         if (value != NULL_PTR) {
1002             switch (sigIdx) {
1003                 case CAN0_SIG_Mod0_temp_valid_0_2:
1004                 case CAN0_SIG_Mod1_temp_valid_0_2:
1005                 case CAN0_SIG_Mod2_temp_valid_0_2:
1006                 case CAN0_SIG_Mod3_temp_valid_0_2:
1007                 case CAN0_SIG_Mod4_temp_valid_0_2:
1008                 case CAN0_SIG_Mod5_temp_valid_0_2:
1009                 case CAN0_SIG_Mod6_temp_valid_0_2:
1010                 case CAN0_SIG_Mod7_temp_valid_0_2:
1011                     if (modIdx >= BS_NR_OF_MODULES) {
1012                         tmpVal = CAN_DEFAULT_VALID_FLAG;
1013                     } else {
1014                         tmp = temp_tab.valid_temperature[modIdx];
1015                     }
1016                     tmpVal = 0x07 & tmp;
```

```
1017                    break;
1018
1019                case CAN0_SIG_Mod0_temp_valid_3_5:
1020                case CAN0_SIG_Mod1_temp_valid_3_5:
1021                case CAN0_SIG_Mod2_temp_valid_3_5:
1022                case CAN0_SIG_Mod3_temp_valid_3_5:
1023                case CAN0_SIG_Mod4_temp_valid_3_5:
1024                case CAN0_SIG_Mod5_temp_valid_3_5:
1025                case CAN0_SIG_Mod6_temp_valid_3_5:
1026                case CAN0_SIG_Mod7_temp_valid_3_5:
1027                    if (modIdx >= BS_NR_OF_MODULES) {
1028                        tmpVal = CAN_DEFAULT_VALID_FLAG;
1029                    } else {
1030                        tmp = temp_tab.valid_temperature[modIdx] >> 3;
1031                    }
1032                    tmpVal = 0x07 & tmp;
1033                    break;
1034
1035                case CAN0_SIG_Mod0_temp_valid_6_8:
1036                case CAN0_SIG_Mod1_temp_valid_6_8:
1037                case CAN0_SIG_Mod2_temp_valid_6_8:
1038                case CAN0_SIG_Mod3_temp_valid_6_8:
1039                case CAN0_SIG_Mod4_temp_valid_6_8:
1040                case CAN0_SIG_Mod5_temp_valid_6_8:
1041                case CAN0_SIG_Mod6_temp_valid_6_8:
1042                case CAN0_SIG_Mod7_temp_valid_6_8:
1043                    if (modIdx >= BS_NR_OF_MODULES) {
1044                        tmpVal = CAN_DEFAULT_VALID_FLAG;
1045                    } else {
1046                        tmp = temp_tab.valid_temperature[modIdx] >> 6;
1047                    }
1048                    tmpVal = 0x07 & tmp;
1049                    break;
1050
1051                case CAN0_SIG_Mod0_temp_valid_9_11:
1052                case CAN0_SIG_Mod1_temp_valid_9_11:
1053                case CAN0_SIG_Mod2_temp_valid_9_11:
1054                case CAN0_SIG_Mod3_temp_valid_9_11:
1055                case CAN0_SIG_Mod4_temp_valid_9_11:
1056                case CAN0_SIG_Mod5_temp_valid_9_11:
1057                case CAN0_SIG_Mod6_temp_valid_9_11:
1058                case CAN0_SIG_Mod7_temp_valid_9_11:
1059                    if (modIdx >= BS_NR_OF_MODULES) {
1060                        tmpVal = CAN_DEFAULT_VALID_FLAG;
1061                    } else {
1062                        tmp = temp_tab.valid_temperature[modIdx] >> 9;
1063                    }
1064                    tmpVal = 0x07 & tmp;
1065                    break;
1066
1067                case CAN0_SIG_Mod0_temp_0:
1068                case CAN0_SIG_Mod0_temp_1:
```

```
1069                case CAN0_SIG_Mod0_temp_2:
1070                case CAN0_SIG_Mod1_temp_0:
1071                case CAN0_SIG_Mod1_temp_1:
1072                case CAN0_SIG_Mod1_temp_2:
1073                case CAN0_SIG_Mod2_temp_0:
1074                case CAN0_SIG_Mod2_temp_1:
1075                case CAN0_SIG_Mod2_temp_2:
1076                case CAN0_SIG_Mod3_temp_0:
1077                case CAN0_SIG_Mod3_temp_1:
1078                case CAN0_SIG_Mod3_temp_2:
1079                case CAN0_SIG_Mod4_temp_0:
1080                case CAN0_SIG_Mod4_temp_1:
1081                case CAN0_SIG_Mod4_temp_2:
1082                case CAN0_SIG_Mod5_temp_0:
1083                case CAN0_SIG_Mod5_temp_1:
1084                case CAN0_SIG_Mod5_temp_2:
1085                case CAN0_SIG_Mod6_temp_0:
1086                case CAN0_SIG_Mod6_temp_1:
1087                case CAN0_SIG_Mod6_temp_2:
1088                case CAN0_SIG_Mod7_temp_0:
1089                case CAN0_SIG_Mod7_temp_1:
1090                case CAN0_SIG_Mod7_temp_2:
1091                    cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
1092                    if ((modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx >= BS_NR_OF_TEMP_SENSORS) {
1093                        tmpVal = CAN_DEFAULT_TEMPERATURE;
1094                    } else {
1095                        tmpVal = temp_tab.temperature[(modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx];
1096                    }
1097                    break;
1098
1099                case CAN0_SIG_Mod0_temp_3:
1100                case CAN0_SIG_Mod0_temp_4:
1101                case CAN0_SIG_Mod0_temp_5:
1102                case CAN0_SIG_Mod1_temp_3:
1103                case CAN0_SIG_Mod1_temp_4:
1104                case CAN0_SIG_Mod1_temp_5:
1105                case CAN0_SIG_Mod2_temp_3:
1106                case CAN0_SIG_Mod2_temp_4:
1107                case CAN0_SIG_Mod2_temp_5:
1108                case CAN0_SIG_Mod3_temp_3:
1109                case CAN0_SIG_Mod3_temp_4:
1110                case CAN0_SIG_Mod3_temp_5:
1111                case CAN0_SIG_Mod4_temp_3:
1112                case CAN0_SIG_Mod4_temp_4:
1113                case CAN0_SIG_Mod4_temp_5:
1114                case CAN0_SIG_Mod5_temp_3:
1115                case CAN0_SIG_Mod5_temp_4:
1116                case CAN0_SIG_Mod5_temp_5:
1117                case CAN0_SIG_Mod6_temp_3:
1118                case CAN0_SIG_Mod6_temp_4:
1119                case CAN0_SIG_Mod6_temp_5:
1120                case CAN0_SIG_Mod7_temp_3:
```

```
1121                case CAN0_SIG_Mod7_temp_4:
1122                case CAN0_SIG_Mod7_temp_5:
1123                    cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
1124                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_3_5 */
1125                    if ((modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx >= BS_NR_OF_TEMP_SENSORS) {
1126                        tmpVal = CAN_DEFAULT_TEMPERATURE;
1127                    } else {
1128                        tmpVal = temp_tab.temperature[(modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx];
1129                    }
1130                    break;
1131
1132                case CAN0_SIG_Mod0_temp_6:
1133                case CAN0_SIG_Mod0_temp_7:
1134                case CAN0_SIG_Mod0_temp_8:
1135                case CAN0_SIG_Mod1_temp_6:
1136                case CAN0_SIG_Mod1_temp_7:
1137                case CAN0_SIG_Mod1_temp_8:
1138                case CAN0_SIG_Mod2_temp_6:
1139                case CAN0_SIG_Mod2_temp_7:
1140                case CAN0_SIG_Mod2_temp_8:
1141                case CAN0_SIG_Mod3_temp_6:
1142                case CAN0_SIG_Mod3_temp_7:
1143                case CAN0_SIG_Mod3_temp_8:
1144                case CAN0_SIG_Mod4_temp_6:
1145                case CAN0_SIG_Mod4_temp_7:
1146                case CAN0_SIG_Mod4_temp_8:
1147                case CAN0_SIG_Mod5_temp_6:
1148                case CAN0_SIG_Mod5_temp_7:
1149                case CAN0_SIG_Mod5_temp_8:
1150                case CAN0_SIG_Mod6_temp_6:
1151                case CAN0_SIG_Mod6_temp_7:
1152                case CAN0_SIG_Mod6_temp_8:
1153                case CAN0_SIG_Mod7_temp_6:
1154                case CAN0_SIG_Mod7_temp_7:
1155                case CAN0_SIG_Mod7_temp_8:
1156                    cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
1157                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_3_5 */
1158                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_6_8 */
1159                    if ((modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx >= BS_NR_OF_TEMP_SENSORS) {
1160                        tmpVal = CAN_DEFAULT_TEMPERATURE;
1161                    } else {
1162                        tmpVal = temp_tab.temperature[(modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx];
1163                    }
1164                    break;
1165
1166                case CAN0_SIG_Mod0_temp_9:
1167                case CAN0_SIG_Mod0_temp_10:
1168                case CAN0_SIG_Mod0_temp_11:
1169                case CAN0_SIG_Mod1_temp_9:
1170                case CAN0_SIG_Mod1_temp_10:
1171                case CAN0_SIG_Mod1_temp_11:
1172                case CAN0_SIG_Mod2_temp_9:
```

```
1173                case CAN0_SIG_Mod2_temp_10:
1174                case CAN0_SIG_Mod2_temp_11:
1175                case CAN0_SIG_Mod3_temp_9:
1176                case CAN0_SIG_Mod3_temp_10:
1177                case CAN0_SIG_Mod3_temp_11:
1178                case CAN0_SIG_Mod4_temp_9:
1179                case CAN0_SIG_Mod4_temp_10:
1180                case CAN0_SIG_Mod4_temp_11:
1181                case CAN0_SIG_Mod5_temp_9:
1182                case CAN0_SIG_Mod5_temp_10:
1183                case CAN0_SIG_Mod5_temp_11:
1184                case CAN0_SIG_Mod6_temp_9:
1185                case CAN0_SIG_Mod6_temp_10:
1186                case CAN0_SIG_Mod6_temp_11:
1187                case CAN0_SIG_Mod7_temp_9:
1188                case CAN0_SIG_Mod7_temp_10:
1189                case CAN0_SIG_Mod7_temp_11:
1190                    cellIdx--;  /* Because cell 0 - valid flag = 1, decrement by one to get the right index */
1191                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_3_5 */
1192                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_6_8 */
1193                    cellIdx--;  /* Because of signal: CAN0_SIG_Modx_temp_valid_9_11 */
1194                    if ((modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx >= BS_NR_OF_TEMP_SENSORS) {
1195                        tmpVal = CAN_DEFAULT_TEMPERATURE;
1196                    } else {
1197                        tmpVal = temp_tab.temperature[(modIdx * BS_NR_OF_TEMP_SENSORS_PER_MODULE) + cellIdx];
1198                    }
1199                    break;
1200
1201                default:
1202                    break;
1203            }
1204            /* Check limits */
1205            canData = cans_checkLimits((float)tmpVal, sigIdx);
1206            /* Apply offset and factor */
1207            *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
            cans_CAN0_signals_tx[sigIdx].factor);
1208        }
1209
1210        return 0;
1211    }
1212
1213
1214    uint32_t cans_gettempering(uint32_t sigIdx, void *value) {          Not implemented.
1215        if (value != NULL_PTR) {
1216            switch (sigIdx) {
1217                case CAN0_SIG_CoolingNeeded:
1218                    *(uint32_t *)value = 0;
1219                    break;
1220
1221                case CAN0_SIG_HeatingNeeded:
1222                    *(uint32_t *)value = 0;
1223                    break;
```

```
1224
1225                case CAN0_SIG_TemperingDemand:
1226                    *(uint32_t *)value = 0;
1227                    break;
1228
1229                default:
1230                    *(uint32_t *)value = 0;
1231                    break;
1232            }
1233        }
1234        return 0;
1235    }
1236
1237
1238    uint32_t cans_getcanerr(uint32_t sigIdx, void *value) {
1239        static DATA_BLOCK_ERRORSTATE_s canerr_tab;
1240        static DATA_BLOCK_MSL_FLAG_s canMSL_tab;
1241        static DATA_BLOCK_RSL_FLAG_s canRSL_tab;
1242        static DATA_BLOCK_MOL_FLAG_s canMOL_tab;
1243        static DATA_BLOCK_CONTFEEDBACK_s cancontfeedback_tab;
1244        static DATA_BLOCK_ILCKFEEDBACK_s canilckfeedback_tab;
1245        static DATA_BLOCK_BALANCING_CONTROL_s balancing_tab;
1246        static DATA_BLOCK_SYSTEMSTATE_s systemstate_tab;
1247
1248        static uint8_t tmp = 0;
1249
1250        if (value != NULL_PTR) {
1251            switch (sigIdx) {
1252                case CAN0_SIG_GS0_general_error:
1253
1254                    /* First signal in CAN_MSG_GeneralState messages -> get database entry */
1255                    DB_ReadBlock(&canerr_tab, DATA_BLOCK_ID_ERRORSTATE);
1256                    DB_ReadBlock(&canMSL_tab, DATA_BLOCK_ID_MSL);
1257                    DB_ReadBlock(&canRSL_tab, DATA_BLOCK_ID_RSL);
1258                    DB_ReadBlock(&canMOL_tab, DATA_BLOCK_ID_MOL);
1259                    tmp = 0;
1260
1261                    /* Check maximum safety limit flags */
1262                    if (canMSL_tab.over_current_charge_cell    == 1 ||
1263                        canMSL_tab.over_current_discharge_cell == 1 ||
1264                        canMSL_tab.over_current_charge_pl0     == 1 ||
1265                        canMSL_tab.over_current_discharge_pl0  == 1 ||
1266                        canMSL_tab.over_current_charge_pl1     == 1 ||
1267                        canMSL_tab.over_current_discharge_pl1  == 1 ||
1268                        canMSL_tab.over_voltage                == 1 ||
1269                        canMSL_tab.under_voltage               == 1 ||
1270                        canMSL_tab.over_temperature_charge     == 1 ||
1271                        canMSL_tab.over_temperature_discharge  == 1 ||
1272                        canMSL_tab.under_temperature_charge    == 1 ||
1273                        canMSL_tab.under_temperature_discharge == 1 ||
1274                    /* Check system error flags */
1275                        canerr_tab.deepDischargeDetected     == 1 ||
```

```
1276                    canerr_tab.main_plus                == 1 ||
1277                    canerr_tab.main_minus               == 1 ||
1278                    canerr_tab.precharge                == 1 ||
1279                    canerr_tab.charge_main_plus         == 1 ||
1280                    canerr_tab.charge_main_minus        == 1 ||
1281                    canerr_tab.charge_precharge         == 1 ||
1282                    canerr_tab.fuse_state_normal        == 1 ||
1283                    canerr_tab.fuse_state_charge        == 1 ||
1284                    canerr_tab.interlock                == 1 ||
1285                    canerr_tab.crc_error                == 1 ||
1286                    canerr_tab.mux_error                == 1 ||
1287                    canerr_tab.spi_error                == 1 ||
1288                    canerr_tab.currentsensorresponding  == 1 ||
1289                    canerr_tab.open_wire                == 1 ||
1290            #if BMS_OPEN_CONTACTORS_ON_INSULATION_ERROR == TRUE
1291                    canerr_tab.insulation_error         == 1 ||
1292            #endif /* BMS_OPEN_CONTACTORS_ON_INSULATION_ERROR */
1293                    canerr_tab.can_timing_cc            == 1 ||
1294                    canerr_tab.can_timing               == 1) {
1295                    /* set flag if error detected */
1296                    tmp |= 0x01 << 0;
1297                }
1298                /* Check recommended safety limit flags */
1299                if (canRSL_tab.over_current_charge_cell    == 1 ||
1300                    canRSL_tab.over_current_discharge_cell == 1 ||
1301                    canRSL_tab.over_current_charge_pl0     == 1 ||
1302                    canRSL_tab.over_current_discharge_pl0  == 1 ||
1303                    canRSL_tab.over_current_charge_pl1     == 1 ||
1304                    canRSL_tab.over_current_discharge_pl1  == 1 ||
1305                    canRSL_tab.over_voltage                == 1 ||
1306                    canRSL_tab.under_voltage               == 1 ||
1307                    canRSL_tab.over_temperature_charge     == 1 ||
1308                    canRSL_tab.over_temperature_discharge  == 1 ||
1309                    canRSL_tab.under_temperature_charge    == 1 ||
1310                    canRSL_tab.under_temperature_discharge == 1) {
1311                    /* set flag if error detected */
1312                    tmp |= 0x01 << 1;
1313                }
1314                /* Check maximum operating limit flags */
1315                if (canMOL_tab.over_current_charge_cell    == 1 ||
1316                    canMOL_tab.over_current_discharge_cell == 1 ||
1317                    canMOL_tab.over_current_charge_pl0     == 1 ||
1318                    canMOL_tab.over_current_discharge_pl0  == 1 ||
1319                    canMOL_tab.over_current_charge_pl1     == 1 ||
1320                    canMOL_tab.over_current_discharge_pl1  == 1 ||
1321                    canMOL_tab.over_voltage                == 1 ||
1322                    canMOL_tab.under_voltage               == 1 ||
1323                    canMOL_tab.over_temperature_charge     == 1 ||
1324                    canMOL_tab.over_temperature_discharge  == 1 ||
1325                    canMOL_tab.under_temperature_charge    == 1 ||
1326                    canMOL_tab.under_temperature_discharge == 1) {
1327                    /* set flag if error detected */
```

```
1328                    tmp |= 0x01 << 2;
1329                }
1330                *(uint32_t *)value = tmp;
1331                break;
1332
1333            case CAN0_SIG_GS0_current_state:
1334                DB_ReadBlock(&systemstate_tab, DATA_BLOCK_ID_SYSTEMSTATE);
1335                *(uint32_t *)value = systemstate_tab.bms_state;
1336                break;
1337            case CAN0_SIG_GS0_error_overtemp_charge:
1338                tmp = 0;
1339                tmp |= canMOL_tab.over_temperature_charge << 2;
1340                tmp |= canRSL_tab.over_temperature_charge << 1;
1341                tmp |= canMSL_tab.over_temperature_charge;
1342                *(uint32_t *)value = tmp;
1343                break;
1344            case CAN0_SIG_GS0_error_undertemp_charge:
1345                tmp = 0;
1346                tmp |= canMOL_tab.under_temperature_charge << 2;
1347                tmp |= canRSL_tab.under_temperature_charge << 1;
1348                tmp |= canMSL_tab.under_temperature_charge;
1349                *(uint32_t *)value = tmp;
1350                break;
1351            case CAN0_SIG_GS0_error_overtemp_discharge:
1352                tmp = 0;
1353                tmp |= canMOL_tab.over_temperature_discharge << 2;
1354                tmp |= canRSL_tab.over_temperature_discharge << 1;
1355                tmp |= canMSL_tab.over_temperature_discharge;
1356                *(uint32_t *)value = tmp;
1357                break;
1358            case CAN0_SIG_GS0_error_undertemp_discharge:
1359                tmp = 0;
1360                tmp |= canMOL_tab.under_temperature_discharge << 2;
1361                tmp |= canRSL_tab.under_temperature_discharge << 1;
1362                tmp |= canMSL_tab.under_temperature_discharge;
1363                *(uint32_t *)value = tmp;
1364                break;
1365            case CAN0_SIG_GS0_error_overcurrent_charge:
1366                tmp = 0;
1367                tmp |= canMOL_tab.over_current_charge_cell << 2;
1368                tmp |= canMOL_tab.over_current_charge_pl0 << 2;
1369                tmp |= canMOL_tab.over_current_charge_pl1 << 2;
1370                tmp |= canRSL_tab.over_current_charge_cell << 1;
1371                tmp |= canRSL_tab.over_current_charge_pl0 << 1;
1372                tmp |= canRSL_tab.over_current_charge_pl1 << 1;
1373                tmp |= canMSL_tab.over_current_charge_cell;
1374                tmp |= canMSL_tab.over_current_charge_pl0;
1375                tmp |= canMSL_tab.over_current_charge_pl1;
1376                *(uint32_t *)value = tmp;
1377                break;
1378            case CAN0_SIG_GS0_error_overcurrent_discharge:
1379                tmp = 0;
```

```
1380                tmp |= canMOL_tab.over_current_discharge_cell << 2;
1381                tmp |= canMOL_tab.over_current_discharge_pl0 << 2;
1382                tmp |= canMOL_tab.over_current_discharge_pl1 << 2;
1383                tmp |= canRSL_tab.over_current_discharge_cell << 1;
1384                tmp |= canRSL_tab.over_current_discharge_pl0 << 1;
1385                tmp |= canRSL_tab.over_current_discharge_pl1 << 1;
1386                tmp |= canMSL_tab.over_current_discharge_cell;
1387                tmp |= canMSL_tab.over_current_discharge_pl0;
1388                tmp |= canMSL_tab.over_current_discharge_pl1;
1389                *(uint32_t *)value = tmp;
1390                break;
1391            case CAN0_SIG_GS1_error_overvoltage:
1392                tmp = 0;
1393                tmp |= canMOL_tab.over_voltage << 2;
1394                tmp |= canRSL_tab.over_voltage << 1;
1395                tmp |= canMSL_tab.over_voltage;
1396                *(uint32_t *)value = tmp;
1397                break;
1398            case CAN0_SIG_GS1_error_undervoltage:
1399                tmp = 0;
1400                tmp |= canMOL_tab.under_voltage << 2;
1401                tmp |= canRSL_tab.under_voltage << 1;
1402                tmp |= canMSL_tab.under_voltage;
1403                *(uint32_t *)value = tmp;
1404                break;
1405            case CAN0_SIG_GS1_error_deep_discharge:
1406                *(uint32_t *)value = canerr_tab.deepDischargeDetected;
1407                break;
1408            case CAN0_SIG_GS1_error_temperature_MCU0:
1409                *(uint32_t *)value = canerr_tab.mcuDieTemperature;
1410                break;
1411            case CAN0_SIG_GS1_error_contactor:
1412                *(uint32_t *)value = canerr_tab.main_plus | canerr_tab.main_minus | canerr_tab.precharge |
                    canerr_tab.charge_main_plus | canerr_tab.charge_main_minus | canerr_tab.charge_precharge;
1413                break;
1414            case CAN0_SIG_GS1_error_selftest:
1415                *(uint32_t *)value = 0;
1416                break;
1417            case CAN0_SIG_GS1_error_cantiming:
1418                *(uint32_t *)value = canerr_tab.can_timing;
1419                break;
1420            case CAN0_SIG_GS1_current_sensor:
1421                *(uint32_t *)value = canerr_tab.currentsensorresponding | canerr_tab.can_timing_cc;
1422                break;
1423            case CAN0_SIG_GS1_balancing_active:
1424
1425                /* only signal to use the balancing database entry */
1426                DB_ReadBlock(&balancing_tab, DATA_BLOCK_ID_BALANCING_CONTROL_VALUES);
1427                *(uint32_t *)value = balancing_tab.enable_balancing;
1428                break;
1429
1430            case CAN0_SIG_GS2_state_cont_interlock:
```

```
1431                    DB_ReadBlock(&cancontfeedback_tab, DATA_BLOCK_ID_CONTFEEDBACK);
1432                    DB_ReadBlock(&canilckfeedback_tab, DATA_BLOCK_ID_ILCKFEEDBACK);
1433                    cancontfeedback_tab.contactor_feedback &= ~(1 << 9);
1434                    cancontfeedback_tab.contactor_feedback |= canilckfeedback_tab.interlock_feedback << 9;
1435                    *(uint32_t *)value = cancontfeedback_tab.contactor_feedback;
1436                    break;
1437
1438                case CAN0_SIG_GS2_error_insulation:
1439                    *(uint32_t *)value = canerr_tab.insulation_error;
1440                    break;
1441
1442                case CAN0_SIG_GS2_fuse_state:
1443                    tmp = 0;
1444                    if (canerr_tab.fuse_state_normal != 0) {
1445    #if BS_CHECK_FUSE_PLACED_IN_NORMAL_PATH == TRUE
1446                        tmp |= 0x01;
1447    #else /* BS_CHECK_FUSE_PLACED_IN_NORMAL_PATH == FALSE */
1448                        tmp |= 0x02;
1449    #endif
1450                    }
1451                    if (canerr_tab.fuse_state_charge != 0) {
1452    #if BS_CHECK_FUSE_PLACED_IN_CHARGE_PATH == TRUE
1453                        tmp |= 0x04;
1454    #else /* BS_CHECK_FUSE_PLACED_IN_CHARGE_PATH == FALSE */
1455                        tmp |= 0x08;
1456    #endif
1457                    }
1458                    *(uint32_t *)value = tmp;
1459                    break;
1460
1461                case CAN0_SIG_GS2_lowCoinCellVolt:
1462                    *(uint32_t *)value = canerr_tab.coinCellVoltage;
1463                    break;
1464
1465                case CAN0_SIG_GS2_error_openWire:
1466                    *(uint32_t *)value = canerr_tab.open_wire;
1467                    break;
1468
1469                case CAN0_SIG_GS2_daisyChain:
1470                    tmp = 0;
1471                    tmp |= canerr_tab.spi_error;
1472                    tmp |= canerr_tab.crc_error << 1;
1473                    tmp |= canerr_tab.mux_error << 2;
1474                    tmp |= canerr_tab.ltc_config_error << 3;
1475                    *(uint32_t *)value = tmp;
1476                    break;
1477
1478                case CAN0_SIG_GS2_plausibilityCheck:
1479                    *(uint32_t *)value = canerr_tab.plausibilityCheck;
1480                    break;
1481
1482                default:
```

```
1483                *(uint32_t *)value = 0;
1484                break;
1485            }
1486        }
1487        return 0;
1488    }
1489
1490
1491    uint32_t cans_getsoc(uint32_t sigIdx, void *value) {
1492        static DATA_BLOCK_SOX_s sox_tab;
1493        DB_ReadBlock(&sox_tab, DATA_BLOCK_ID_SOX);
1494        if (value != NULL_PTR) {
1495            switch (sigIdx) {
1496                case CAN0_SIG_SOC_mean:
1497                    /* CAN signal resolution 0.01%, --> factor 100 */
1498                    *(uint32_t *)value = (uint32_t)(sox_tab.soc_mean * cans_CAN0_signals_tx[sigIdx].factor);
1499                    break;
1500                case CAN0_SIG_SOC_min:
1501                    /* CAN signal resolution 0.01%, --> factor 100 */
1502                    *(uint32_t *)value = (uint32_t)(sox_tab.soc_min * cans_CAN0_signals_tx[sigIdx].factor);
1503                    break;
1504                case CAN0_SIG_SOC_max:
1505                    /* CAN signal resolution 0.01%, --> factor 100 */
1506                    *(uint32_t *)value = (uint32_t)(sox_tab.soc_max * cans_CAN0_signals_tx[sigIdx].factor);
1507                    break;
1508                default:
1509                    *(uint32_t *)value = 50.0;
1510                    break;
1511            }
1512        }
1513        return 0;
1514    }
1515
1516
1517    static uint32_t cans_getRecommendedOperatingCurrent(uint32_t sigIdx, void *value) {
1518        static DATA_BLOCK_SOF_s sof_tab;
1519        float canData = 0;
1520
1521        if (value != NULL_PTR) {
1522            /* values transmitted in resolution of 10mA (16bit means 0A-655.35A) */
1523            switch (sigIdx) {
1524                case CAN0_SIG_RecChargeCurrent:
1525                    /* first signal */
1526                    DB_ReadBlock(&sof_tab, DATA_BLOCK_ID_SOF);
1527
1528                    /* Check limits */
1529                    canData = cans_checkLimits((float)sof_tab.recommended_continuous_charge, sigIdx);
1530                    /* Apply offset and factor */
1531                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1532                    break;
1533
```

```
1534                case CAN0_SIG_RecChargeCurrent_Peak:
1535                    /* Check limits */
1536                    canData = cans_checkLimits((float)sof_tab.recommended_peak_charge, sigIdx);
1537                    /* Apply offset and factor */
1538                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1539                    break;
1540
1541                case CAN0_SIG_RecDischargeCurrent:
1542                    /* Check limits */
1543                    canData = cans_checkLimits((float)sof_tab.recommended_continuous_discharge, sigIdx);
1544                    /* Apply offset and factor */
1545                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1546                    break;
1547
1548                case CAN0_SIG_RecDischargeCurrent_Peak:
1549                    /* Check limits */
1550                    canData = cans_checkLimits((float)sof_tab.recommended_peak_discharge, sigIdx);
1551                    /* Apply offset and factor */
1552                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1553                    break;
1554
1555                default:
1556                    break;
1557            }
1558        }
1559        return 0;
1560    }
1561
1562
1563    static uint32_t cans_getMaxAllowedPower(uint32_t sigIdx, void *value) {
1564        if (value != NULL_PTR) {
1565            switch (sigIdx) {
1566                default:
1567                    *(uint32_t *)value = 0;
1568                    break;
1569            }
1570        }
1571        return 0;
1572    }
1573
1574
1575    static uint32_t cans_getminmaxvolt(uint32_t sigIdx, void *value) {
1576        static DATA_BLOCK_MINMAX_s minmax_volt_tab;
1577        float canData = 0;
1578
1579        if (value != NULL_PTR) {
1580            switch (sigIdx) {
1581                case CAN0_SIG_Cellvolt_mean:
1582                    /* First signal that is called */
```

```
1583                    DB_ReadBlock(&minmax_volt_tab, DATA_BLOCK_ID_MINMAX);
1584
1585                    /* Check limits */
1586                    canData = cans_checkLimits((float)minmax_volt_tab.voltage_mean, sigIdx);
1587                    /* Apply offset and factor */
1588                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1589                    break;
1590
1591                case CAN0_SIG_Cellvolt_min:
1592                    /* Check limits */
1593                    canData = cans_checkLimits((float)minmax_volt_tab.voltage_min, sigIdx);
1594                    /* Apply offset and factor */
1595                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1596                    break;
1597
1598                case CAN0_SIG_Cellvolt_max:
1599                    /* Check limits */
1600                    canData = cans_checkLimits((float)minmax_volt_tab.voltage_max, sigIdx);
1601                    /* Apply offset and factor */
1602                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1603                    break;
1604
1605                case CAN0_SIG_ModNumber_volt_min:
1606                    /* Check limits */
1607                    canData = cans_checkLimits((float)minmax_volt_tab.voltage_module_number_min, sigIdx);
1608                    /* Apply offset and factor */
1609                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1610                    break;
1611
1612                case CAN0_SIG_ModNumber_volt_max:
1613                    /* Check limits */
1614                    canData = cans_checkLimits((float)minmax_volt_tab.voltage_module_number_max, sigIdx);
1615                    /* Apply offset and factor */
1616                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1617                    break;
1618
1619                default:
1620                    *(uint32_t *)value = 0;
1621                    break;
1622            }
1623        }
1624        return 0;
1625    }
1626
1627    uint32_t cans_getminmaxtemp(uint32_t sigIdx, void *value) {
1628        static DATA_BLOCK_MINMAX_s minmax_temp_tab;
1629        float canData = 0;
```

```
1630
1631        if (value != NULL_PTR) {
1632            switch (sigIdx) {
1633            case CAN0_SIG_Celltemp_mean:
1634                /* First signal that is called */
1635                DB_ReadBlock(&minmax_temp_tab, DATA_BLOCK_ID_MINMAX);
1636
1637                /*  Check limits */
1638                canData = cans_checkLimits((float)minmax_temp_tab.temperature_mean, sigIdx);
1639                /* Apply offset and factor */
1640                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1641                break;
1642
1643            case CAN0_SIG_Celltemp_min:
1644                /* Check limits */
1645                canData = cans_checkLimits((float)minmax_temp_tab.temperature_min, sigIdx);
1646                /* Apply offset and factor */
1647                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1648                break;
1649
1650            case CAN0_SIG_Celltemp_max:
1651                /* Check limits */
1652                canData = cans_checkLimits((float)minmax_temp_tab.temperature_max, sigIdx);
1653                /* Apply offset and factor */
1654                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1655                break;
1656
1657            case CAN0_SIG_ModNumber_temp_min:
1658                /* Check limits */
1659                canData = cans_checkLimits((float)minmax_temp_tab.temperature_module_number_min, sigIdx);
1660                /* Apply offset and factor */
1661                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1662                break;
1663
1664            case CAN0_SIG_ModNumber_temp_max:
1665                /* Check limits */
1666                canData = cans_checkLimits((float)minmax_temp_tab.temperature_module_number_max, sigIdx);
1667                /* Apply offset and factor */
1668                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1669                break;
1670
1671            default:
1672                *(uint32_t *)value = 0;
1673                break;
1674            }
1675        }
1676        return 0;
```

```
1677  }
1678
1679
1680  #ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
1681  uint32_t cans_gettriggercurrent(uint32_t sigIdx, void *value) {
1682      *(uint32_t *)value = 0x00FFFF31;
1683      return 0;
1684  }
1685  #endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
1686
1687
1688  static uint32_t cans_getpower(uint32_t sigIdx, void *value) {
1689      uint32_t retVal = 0;
1690      float canData = 0;
1691      static DATA_BLOCK_MOVING_AVERAGE_s powMovMean_tab;
1692
1693      if (value != NULL_PTR) {
1694         switch (sigIdx) {
1695             case CAN0_SIG_MovAverage_Power_1s:
1696                 /* first signal to call function */
1697                 DB_ReadBlock(&powMovMean_tab, DATA_BLOCK_ID_MOV_AVERAGE);
1698                 /* Check limits */
1699                 canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_1s, sigIdx);
1700                 /* Apply offset and factor */
1701                 *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1702                 break;
1703
1704             case CAN0_SIG_MovAverage_Power_5s:
1705                 /* Check limits */
1706                 canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_5s, sigIdx);
1707                 /* Apply offset and factor */
1708                 *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1709                 break;
1710
1711             case CAN0_SIG_MovAverage_Power_10s:
1712                 /* Check limits */
1713                 canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_10s, sigIdx);
1714                 /* Apply offset and factor */
1715                 *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1716                 break;
1717
1718             case CAN0_SIG_MovAverage_Power_30s:
1719                 /* Check limits */
1720                 canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_30s, sigIdx);
1721                 /* Apply offset and factor */
1722                 *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1723                 break;
1724
```

```
1725                case CAN0_SIG_MovAverage_Power_60s:
1726                    /* Check limits */
1727                    canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_60s, sigIdx);
1728                    /* Apply offset and factor */
1729                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1730                    break;
1731
1732                case CAN0_SIG_MovAverage_Power_config:
1733                    /* Check limits */
1734                    canData = cans_checkLimits((float)powMovMean_tab.movAverage_power_config, sigIdx);
1735                    /* Apply offset and factor */
1736                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1737                    break;
1738
1739                default:
1740                    *(uint32_t *)value = 0;
1741                    break;
1742            }
1743        }
1744        return retVal;
1745    }
1746
1747    static uint32_t cans_getcurr(uint32_t sigIdx, void *value) {
1748        uint32_t retVal = 0;
1749        float canData = 0;
1750        static DATA_BLOCK_MOVING_AVERAGE_s curMovMean_tab;
1751
1752        if (value != NULL_PTR) {
1753           switch (sigIdx) {
1754                case CAN0_SIG_MovAverage_Current_1s:
1755                    /* first signal to call function */
1756                    DB_ReadBlock(&curMovMean_tab, DATA_BLOCK_ID_MOV_AVERAGE);
1757                    /* Check limits */
1758                    canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_1s, sigIdx);
1759                    /* Apply offset and factor */
1760                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1761                    break;
1762
1763                case CAN0_SIG_MovAverage_Current_5s:
1764                    /* Check limits */
1765                    canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_5s, sigIdx);
1766                    /* Apply offset and factor */
1767                    *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                        cans_CAN0_signals_tx[sigIdx].factor);
1768                    break;
1769
1770                case CAN0_SIG_MovAverage_Current_10s:
1771                    /* Check limits */
1772                    canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_10s, sigIdx);
```

```
1773                /* Apply offset and factor */
1774                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1775                break;
1776
1777            case CAN0_SIG_MovAverage_Current_30s:
1778                /* Check limits */
1779                canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_30s, sigIdx);
1780                /* Apply offset and factor */
1781                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1782                break;
1783
1784            case CAN0_SIG_MovAverage_Current_60s:
1785                /* Check limits */
1786                canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_60s, sigIdx);
1787                /* Apply offset and factor */
1788                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1789                break;
1790
1791            case CAN0_SIG_MovAverage_Current_config:
1792                /* Check limits */
1793                canData = cans_checkLimits((float)curMovMean_tab.movAverage_current_config, sigIdx);
1794                /* Apply offset and factor */
1795                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1796                break;
1797
1798            default:
1799                    *(uint32_t *)value = 0;
1800                    break;
1801        }
1802    }
1803    return retVal;
1804 }
1805
1806
1807 static uint32_t cans_getPackVoltage(uint32_t sigIdx, void *value) {
1808    uint32_t retVal = 0;
1809    float canData = 0;
1810    static DATA_BLOCK_CURRENT_SENSOR_s packVolt_tab;
1811
1812    if (value != NULL_PTR) {
1813       switch (sigIdx) {
1814            case CAN0_SIG_PackVolt_Battery:
1815                /* first signal to call function */
1816                DB_ReadBlock(&packVolt_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1817                /* Check limits */
1818                canData = cans_checkLimits((float)packVolt_tab.voltage[0], sigIdx);
1819                /* Apply offset and factor */
1820                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
```

```
                cans_CAN0_signals_tx[sigIdx].factor);
1821                break;
1822
1823            case CAN0_SIG_PackVolt_PowerNet:
1824                /* first signal to call function */
1825                DB_ReadBlock(&packVolt_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1826                /* Check limits */
1827                canData = cans_checkLimits((float)packVolt_tab.voltage[2], sigIdx);
1828                /* Apply offset and factor */
1829                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                cans_CAN0_signals_tx[sigIdx].factor);
1830                break;
1831
1832            default:
1833                break;
1834        }
1835    }
1836    return retVal;
1837 }
1838
1839
1840 static uint32_t cans_setcurr(uint32_t sigIdx, void *value) {
1841     int32_t currentValue;
1842     int32_t temperatureValue;
1843     int32_t powerValue;
1844     int32_t currentcounterValue;
1845     int32_t energycounterValue;
1846     int32_t voltageValue[3];
1847     uint32_t idx = 0;
1848     uint8_t dummy = 0;
1849
1850     if (value != NULL_PTR) {
1851         switch (sigIdx) {
1852             case CAN0_SIG_IVT_Current_Status:
1853             case CAN0_SIG_IVT_Voltage_1_Status:
1854             case CAN0_SIG_IVT_Voltage_2_Status:
1855             case CAN0_SIG_IVT_Voltage_3_Status:
1856             case CAN0_SIG_IVT_Temperature_Status:
1857             case CAN0_SIG_IVT_Power_Status:
1858             case CAN0_SIG_IVT_CC_Status:
1859             case CAN0_SIG_IVT_EC_Status:
1860                 dummy = *(uint32_t *)value & 0x000000FF;
1861                 dummy &= 0xF0;   /* only high nibble contains diag info */
1862                 if ((dummy & 0x10) == TRUE) {
1863                     /* Overcurrent detected. This feature is currently not supported. */
1864                 }
1865                 if ((dummy & 0x20) == TRUE) {
1866                     if (sigIdx == CAN0_SIG_IVT_Current_Status) {
1867                         cans_current_tab.state_current = 1;
1868                     } else if (sigIdx == CAN0_SIG_IVT_Voltage_1_Status || sigIdx == CAN0_SIG_IVT_Voltage_2_Status ||
                    sigIdx == CAN0_SIG_IVT_Voltage_3_Status) {
1869                         cans_current_tab.state_voltage = 1;
```

```
1870                    } else if (sigIdx == CAN0_SIG_IVT_Temperature_Status) {
1871                        cans_current_tab.state_temperature = 1;
1872                    } else if (sigIdx == CAN0_SIG_IVT_Power_Status) {
1873                        cans_current_tab.state_power = 1;
1874                    } else if (sigIdx == CAN0_SIG_IVT_CC_Status) {
1875                        cans_current_tab.state_cc = 1;
1876                    } else {
1877                        cans_current_tab.state_ec = 1;
1878                    }
1879                } else {
1880                    cans_current_tab.state_current = 0;
1881                    cans_current_tab.state_voltage = 0;
1882                    cans_current_tab.state_temperature = 0;
1883                    cans_current_tab.state_power = 0;
1884                    cans_current_tab.state_cc = 0;
1885                    cans_current_tab.state_ec = 0;
1886                }
1887                if ((dummy & 0x40) == TRUE || (dummy & 0x80) == TRUE) {
1888                    cans_current_tab.state_current = 1;
1889                    cans_current_tab.state_voltage = 1;
1890                    cans_current_tab.state_temperature = 1;
1891                    cans_current_tab.state_power = 1;
1892                    cans_current_tab.state_cc = 1;
1893                    cans_current_tab.state_ec = 1;
1894                }
1895
1896                break;
1897
1898                case CAN0_SIG_IVT_Current_Measurement:
1899                /* case CAN1_SIG_ISENS0_I_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                signal struct */
1900                    currentValue = *(int32_t*)value;
1901                    cans_current_tab.current = (currentValue);
1902                    cans_current_tab.newCurrent++;
1903                    cans_current_tab.previous_timestamp_cur = cans_current_tab.timestamp_cur;
1904                    cans_current_tab.timestamp_cur = OS_getOSSysTick();
1905                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1906                    break;
1907                case CAN0_SIG_IVT_Voltage_1_Measurement:
1908                /* case CAN1_SIG_ISENS1_U1_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                signal struct */
1909                    idx = 0;
1910                    voltageValue[idx] = *(int32_t*)value;
1911                    cans_current_tab.voltage[idx] = (float)(voltageValue[idx])*cans_CAN0_signals_rx[sigIdx].factor;
1912                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1913                    break;
1914                case CAN0_SIG_IVT_Voltage_2_Measurement:
1915                /* case CAN1_SIG_ISENS2_U2_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                signal struct */
1916                    idx = 1;
1917                    voltageValue[idx] = *(int32_t*)value;
1918                    cans_current_tab.voltage[idx] = (float)(voltageValue[idx])*cans_CAN0_signals_rx[sigIdx].factor;
```

```
1919                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1920                    break;
1921                case CAN0_SIG_IVT_Voltage_3_Measurement:
1922                /* case CAN1_SIG_ISENS3_U3_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                    signal struct */
1923                    idx = 2;
1924                    voltageValue[idx] = *(int32_t*)value;
1925                    cans_current_tab.voltage[idx]=(float)(voltageValue[idx])*cans_CAN0_signals_rx[sigIdx].factor;
1926                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1927                    break;
1928                case CAN0_SIG_IVT_Temperature_Measurement:
1929                /* case CAN1_SIG_ISENS4_T_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                    signal struct */
1930                    temperatureValue = *(int32_t*)value;
1931                    cans_current_tab.temperature = (float)(temperatureValue)*cans_CAN0_signals_rx[sigIdx].factor;
1932                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1933                    break;
1934                case CAN0_SIG_IVT_Power_Measurement:
1935                /* case CAN1_SIG_ISENS5_P_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                    signal struct */
1936                    powerValue = *(int32_t*)value;
1937                    cans_current_tab.power = (float)(powerValue);
1938                    cans_current_tab.newPower++;
1939                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1940                    break;
1941                case CAN0_SIG_IVT_CC_Measurement:
1942                /* case CAN1_SIG_ISENS6_CC_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                    signal struct */
1943                    currentcounterValue = *(int32_t*)value;
1944                    cans_current_tab.previous_timestamp_cc = cans_current_tab.timestamp_cc;
1945                    cans_current_tab.timestamp_cc = OS_getOSSysTick();
1946                    cans_current_tab.current_counter = (float)(currentcounterValue);
1947                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1948                    break;
1949                case CAN0_SIG_IVT_EC_Measurement:
1950                /* case CAN1_SIG_ISENS7_EC_Measurement:  uncommented because identical position in CAN0 and CAN1 rx
                    signal struct */
1951                    energycounterValue = *(int32_t*)value;
1952                    cans_current_tab.energy_counter = (float)(energycounterValue);
1953                    DB_WriteBlock(&cans_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);
1954                    break;
1955            }
1956        }
1957        return 0;
1958    }
1959
1960
1961    uint32_t cans_setstaterequest(uint32_t sigIdx, void *value) {    This is the callback function for setting state request for BMS/CONT.
1962        DATA_BLOCK_STATEREQUEST_s staterequest_tab;
1963        uint8_t staterequest;
1964
1965        DB_ReadBlock(&staterequest_tab, DATA_BLOCK_ID_STATEREQUEST);
```

```
1966
1967        if (value != NULL_PTR) {
1968            if (sigIdx == CAN0_SIG_ReceiveStateRequest) {
1969                staterequest = *(uint8_t *)value;
1970                staterequest_tab.previous_state_request = staterequest_tab.state_request;
1971                staterequest_tab.state_request = staterequest;
1972                if ((staterequest_tab.state_request != staterequest_tab.previous_state_request)|| \
1973                        (OS_getOSSysTick()- staterequest_tab.timestamp) > 3000) {
1974                    staterequest_tab.state_request_pending = staterequest;
1975                }
1976                staterequest_tab.state++;
1977                DB_WriteBlock(&staterequest_tab, DATA_BLOCK_ID_STATEREQUEST);
1978            }
1979        }
1980        return 0;
1981    }
1982
1983
1984    uint32_t cans_getisoguard(uint32_t sigIdx, void *value) {
1985        static DATA_BLOCK_ISOMETER_s isoguard_tab;
1986        float canData = 0;
1987
1988        if (value != NULL_PTR) {
1989            switch (sigIdx) {
1990                case CAN0_SIG_InsulationStatus:
1991                /* First signal call */
1992                DB_ReadBlock(&isoguard_tab, DATA_BLOCK_ID_ISOGUARD);
1993
1994                /* Check limits */
1995                canData = cans_checkLimits((float)isoguard_tab.state, sigIdx);
1996                /* Apply offset and factor */
1997                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
1998                break;
1999
2000                case CAN0_SIG_InsulationValue:
2001                /* Check limits */
2002                canData = cans_checkLimits((float)isoguard_tab.resistance_kOhm, sigIdx);
2003                /* Apply offset and factor */
2004                *(uint32_t *)value = (uint32_t)((canData + cans_CAN0_signals_tx[sigIdx].offset) *
                    cans_CAN0_signals_tx[sigIdx].factor);
2005                break;
2006
2007                default:
2008                    *(uint32_t *)value = 0;
2009                    break;
2010            }
2011        }
2012        return 0;
2013    }
2014
2015
```

```
2016  uint32_t cans_setdebug(uint32_t sigIdx, void *value) {
2017      uint8_t data[8] = {0, 0, 0, 0, 0, 0, 0, 0};
2018      DATA_BLOCK_SYSTEMSTATE_s systemstate_tab;
2019
2020
2021      data[0] = *(uint64_t *)value & 0x00000000000000FF;
2022      data[1] = (*(uint64_t *)value & 0x000000000000FF00) >> 8;
2023      data[2] = (*(uint64_t *)value & 0x0000000000FF0000) >> 16;
2024      data[3] = (*(uint64_t *)value & 0x00000000FF000000) >> 24;
2025      data[4] = (*(uint64_t *)value & 0x000000FF00000000) >> 32;
2026      data[5] = (*(uint64_t *)value & 0x0000FF0000000000) >> 40;
2027      data[6] = (*(uint64_t *)value & 0x00FF000000000000) >> 48;
2028      data[7] = (*(uint64_t *)value & 0xFF00000000000000) >> 56;
2029
2030
2031      if (value != NULL_PTR) {
2032          switch (data[0]) {
2033              case 0x0B:  /* Set Soc directly with value. Unit in CAN message: 0.01 percent --> range 0...10000 means
                  0%Soc...100%Soc */
2034                  SOC_SetValue(((float)((data[1]) << 8 | (data[2])))/100.0f, ((float)((data[1]) << 8 |
                      (data[2])))/100.0f, ((float)((data[1]) << 8 | (data[2])))/100.0f); /* divide by 100 to get SOC
                      between 0 and 100 */
2035                  break;
2036              case 0x0E:  /* debug Message for Balancing on pack level */
2037                  if (data[1] == 0) {
2038                      BAL_SetStateRequest(BAL_STATE_GLOBAL_DISABLE_REQUEST);
2039                  } else {
2040                      BAL_SetStateRequest(BAL_STATE_GLOBAL_ENABLE_REQUEST);
2041                  }
2042                  break;
2043              case 0xAA:
2044                  DIAG_Handler(DIAG_CH_DEEP_DISCHARGE_DETECTED, DIAG_EVENT_OK, 0);
2045                  break;
2046              case 0xBB:
2047                  DB_ReadBlock(&systemstate_tab, DATA_BLOCK_ID_SYSTEMSTATE);
2048                  systemstate_tab.bms_state = 0xF0;
2049                  DB_WriteBlock(&systemstate_tab, DATA_BLOCK_ID_SYSTEMSTATE);
2050                  break;
2051              default:
2052                  break;
2053          }
2054      }
2055      return 0;
2056  }
2057
2058
2059  uint32_t cans_setSWversion(uint32_t sigIdx, void *value) {
2060      SYS_SendBootMessage(0);
2061      return 0;
2062  }
2063
2064  static uint32_t cans_setenginerequest(uint32_t sidIdx, void *value) {
```

```
2065        uint8_t data = *(uint64_t *)value & 0xFF;
2066
2067        if (data == 1) {
2068            // Close contactor
2069            // CONT_SetContactorState(CONT_ENGINE, CONT_SWITCH_ON);
2070        } else {
2071            // Open contactor
2072            // CONT_SetContactorState(CONT_ENGINE, CONT_SWITCH_OFF);
2073        }
2074
2075        return 0;
2076    }
2077
2078
2079    float cans_checkLimits(float value, uint32_t sigIdx) {
2080        float retVal = value;
2081
2082        if (value < cans_CAN0_signals_tx[sigIdx].min)
2083            retVal = cans_CAN0_signals_tx[sigIdx].min;
2084
2085        if (value > cans_CAN0_signals_tx[sigIdx].max)
2086            retVal = cans_CAN0_signals_tx[sigIdx].max;
2087
2088        return retVal;
2089    }
2090
2091    /*================== Extern Function Implementations ========================*/
2092
```