can\_cfg\_tx-message-definitions.h

AlarmWarningsBattery (359h)

Add:

/\*\* CAN message properties for BMS state message. Required properties are:

 \* - Message ID

 \* - Identifier type (standard or extended)

 \* - Message period and phase in ms

 \* - Endianness of message data @{\*/

#define ALARMS\_WARNINGS\_BATTERY\_ID         (0x359u)

#define ALARMS\_WARNINGS\_BATTERY\_ID\_TYPE    (CAN\_STANDARD\_IDENTIFIER\_11\_BIT)

#define ALARMS\_WARNINGS\_BATTERY\_PERIOD\_ms  (100u)

#define ALARMS\_WARNINGS\_BATTERY\_PHASE\_ms   (0u)

#define ALARMS\_WARNINGS\_BATTERY\_ENDIANNESS (CAN\_BIG\_ENDIAN)

/\*\*@}\*/

#define ALARMS\_WARNINGS\_BATTERY\_MESSAGE                                                \

    {                                                                          \

        .id         = ALARMS\_WARNINGS\_BATTERY\_ID,                                      \

        .idType     = ALARMS\_WARNINGS\_BATTERY\_ID\_TYPE,                                 \

        .dlc        = CAN\_DEFAULT\_DLC,                                         \

        .endianness = ALARMS\_WARNINGS\_BATTERY\_ENDIANNESS,                              \

    },                                                                         \

    {                                                                          \

        .period = ALARMS\_WARNINGS\_BATTERY\_PERIOD\_ms, .phase = ALARMS\_WARNINGS\_BATTERY\_PHASE\_ms \

    }

BatteryLimits (351h)

Add:

/\*\* CAN message properties for BMS limit values. Required properties are:

 \* - Message ID

 \* - Identifier type (standard or extended)

 \* - Message period and phase in ms

 \* - Endianness of message data @{\*/

#define BATTERY\_LIMITS\_ID         (0x351u)

#define BATTERY\_LIMITS\_ID\_TYPE    (CAN\_STANDARD\_IDENTIFIER\_11\_BIT)

#define BATTERY\_LIMITS\_PERIOD\_ms  (100u)

#define BATTERY\_LIMITS\_PHASE\_ms   (30u)

#define BATTERY\_LIMITS\_ENDIANNESS (CAN\_BIG\_ENDIAN)

/\*\*@}\*/

#define BATTERY\_LIMITS\_MESSAGE                                                   \

    {                                                                                \

        .id         = BATTERY\_LIMITS\_ID,                                         \

        .idType     = BATTERY\_LIMITS\_ID\_TYPE,                                    \

        .dlc        = CAN\_DEFAULT\_DLC,                                               \

        .endianness = BATTERY\_LIMITS\_ENDIANNESS,                                 \

    },                                                                               \

    {                                                                                \

        .period = BATTERY\_LIMITS\_PERIOD\_ms, .phase = BATTERY\_LIMITS\_PHASE\_ms \

    }

can\_cfg\_tx.c

BatteryLimits (351h)

Add:

{CAN\_NODE\_1, BATTERY\_LIMITS\_MESSAGE, &BatteryLimits, NULL\_PTR},

can\_cbs\_tx.h

BatteryLimits (351h)

Add:

/\*\*

 \* @brief can tx callback function for limit values

 \* @param[in] message     contains the message ID, DLC and endianness

 \* @param[in] pCanData    payload of can frame

 \* @param[in] pMuxId      multiplexer for multiplexed CAN messages

 \* @param[in] kpkCanShim  shim to the database entries

 \*/

extern uint32\_t BatteryLimits(

    CAN\_MESSAGE\_PROPERTIES\_s message,

    uint8\_t \*pCanData,

    uint8\_t \*pMuxId,

    const CAN\_SHIM\_s \*const kpkCanShim);

can\_cbs\_tx\_limit-values.c

BatteryLimits (351h)

Add:

extern uint32\_t BatteryLimits(

    CAN\_MESSAGE\_PROPERTIES\_s message,

    uint8\_t \*pCanData,

    uint8\_t \*pMuxId,

    const CAN\_SHIM\_s \*const kpkCanShim) {

    /\* pMuxId is not used here, therefore has to be NULL\_PTR \*/

    FAS\_ASSERT(pMuxId == NULL\_PTR);

    FAS\_ASSERT(message.id == BATTERY\_LIMITS\_ID);

    FAS\_ASSERT(message.idType == BATTERY\_LIMITS\_ID\_TYPE);

    FAS\_ASSERT(message.dlc == CAN\_FOXBMS\_MESSAGES\_DEFAULT\_DLC);

    FAS\_ASSERT(pCanData != NULL\_PTR);

    FAS\_ASSERT(kpkCanShim != NULL\_PTR);

    uint64\_t messageData = 0u;

    DATA\_READ\_DATA(kpkCanShim->pTableSof);

    /\* AXIVION Disable Style Generic-NoMagicNumbers: Signal data defined in .dbc file. \*/

    /\* maximum charge current \*/

    float\_t signalData = (float\_t)kpkCanShim->pTableSof->recommendedContinuousPackChargeCurrent\_mA;

    float\_t offset     = 0.0f;

    float\_t factor     = 0.004f; /\* convert mA to 250mA \*/

    signalData         = (signalData + offset) \* factor;

    uint64\_t data      = (int64\_t)signalData;

    /\* set data in CAN frame \*/

    CAN\_TxSetMessageDataWithSignalData(&messageData, 23u, 16u, data, message.endianness);

    /\* maximum discharge current \*/

    signalData = (float\_t)kpkCanShim->pTableSof->recommendedContinuousPackDischargeCurrent\_mA;

    offset     = 0.0f;

    factor     = 0.004f; /\* convert mA to 250mA \*/

    signalData = (signalData + offset) \* factor;

    data       = (int64\_t)signalData;

    /\* set data in CAN frame \*/

    CAN\_TxSetMessageDataWithSignalData(&messageData, 39u, 16u, data, message.endianness);

    /\* maximum pack voltage \*/

    signalData = (float\_t)(BS\_NR\_OF\_CELL\_BLOCKS\_PER\_MODULE \* BC\_VOLTAGE\_MAX\_MOL\_mV);

    offset     = 0.0f;

    factor     = 0.01f; /\* convert mV to 1V \*/

    signalData = (signalData + offset) \* factor;

    data       = (int64\_t)signalData;

    /\* set data in CAN frame \*/

    CAN\_TxSetMessageDataWithSignalData(&messageData, 7u, 16u, data, message.endianness);

    /\* AXIVION Enable Style Generic-NoMagicNumbers: \*/

    /\* now copy data in the buffer that will be used to send data \*/

    CAN\_TxSetCanDataWithMessageData(messageData, pCanData, message.endianness);

    return 0;

}