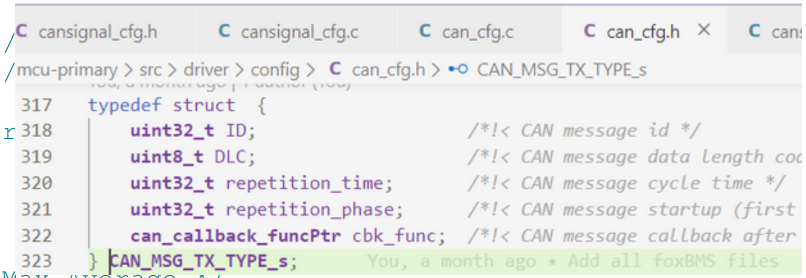
```c
/**
 *
 * @copyright &copy; 2010 - 2020, Fraunhofer-Gesellschaft zur Foerderung der
 *  angewandten Forschung e.V. All rights reserved.
 *
 * BSD 3-Clause License
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions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

/**
 * @file    can_cfg.c
 * @author  foxBMS Team
 * @date    12.07.2015 (date of creation)
 * @ingroup DRIVERS_CONF
 * @prefix  CAN
 *
 * @brief   Configuration for the CAN module
 *
 * The CAN bus settings and the received messages and their
 * reception handling are to be specified here.
 *
 */

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

#include "batterysystem_cfg.h"
#include "mcu.h"
#include "rcc_cfg.h"

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

/*================== Constant and Variable Definitions =====================*/

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

/*================== Function Implementations ==============================*/

/* ***************************************
 *  Set CAN settings here
 *****************************************/

CAN_HandleTypeDef hcan0 = {
        .Instance = CAN2,
        .State = HAL_CAN_STATE_RESET,
        .ErrorCode = HAL_CAN_ERROR_NONE,
#if (CAN0_BAUDRATE == 1000000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 3,        /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/3/15 = 1.0MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 3,        /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/3/14 = 1.0MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 4,        /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/4/8 = 1.0MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN0_BAUDRATE == 500000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 6,        /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/6/15 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 6,        /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/6/14 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 8,        /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/8/8 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN0_BAUDRATE == 250000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 12,       /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/12/15 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 12,       /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/12/14 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 16,       /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_2TQ,    /* --> CAN = 32MHz/16/8 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_5TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN0_BAUDRATE == 125000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 24,       /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/12/14 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 24,       /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/12/14 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 32,       /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_2TQ,    /* --> CAN = 32MHz/16/8 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_5TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#endif
        .Init.Mode = CAN_MODE_NORMAL,  /* for test purpose, without connected can-bus use LOOPBACK mode */
        .Init.SyncJumpWidth = CAN_SJW_1TQ,
        .Init.TimeTriggeredMode = DISABLE,   /* time triggerd communication mode */
                            /* DISABLE: no influence */
                            /* ENABLE: saves timestamps for received and transmitted messages. See reference manual for more information. */
                            /* DISABLE: Manually re-initialize CAN and wait for 128 * 11 recessive bits */
        .Init.AutoBusOff = ENABLE,    /* automatic bus-off management */
                            /* ENABLE: automatically leave bus-off mode after 128 * 11 recessive bits */
        .Init.AutoWakeUp = ENABLE,    /* automatic wake-up mode */
                            /* ENABLE: automatically leave sleep mode on message receiving */
                            /* DISABLE: SLEEP bit needs to be deleted by software */
        .Init.AutoRetransmission = DISABLE,   /* automatic retransition mode; */
                            /* DISABLE: retransmit the message until it has been successfully transmitted */
                            /* ENABLE: transmit only once, independently of transmission result */
        .Init.ReceiveFifoLocked = ENABLE,    /* Receive FIFO locked against overrun. */
                            /* DISABLE: A new incoming message overwrites the last received message. */
                            /* ENABLE: Once a receive FIFO is full the next incoming message will be discarded. */
        .Init.TransmitFifoPriority = ENABLE,   /* Transmit FIFO priority */
                            /* DISABLE: driven by identifier of message. Lower identifier equals higher priority */
                            /* ENABLE: driven chronologically */
};



CAN_HandleTypeDef hcan1 = {
        .Instance = CAN1,
        .State = HAL_CAN_STATE_RESET,
        .ErrorCode = HAL_CAN_ERROR_NONE,
#if (CAN1_BAUDRATE == 1000000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 3,        /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/3/15 = 1.0MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 3,        /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/3/14 = 1.0MHz */
```

```c
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 4,        /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/4/8 = 1.0MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN1_BAUDRATE == 500000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 6,        /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/6/15 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 6,        /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/6/14 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 8,        /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/8/8 = 0.5MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN1_BAUDRATE == 250000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 12,       /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/12/15 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 12,       /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/12/14 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 16,       /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/16/8 = 0.25MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
```

```
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#elif(CAN1_BAUDRATE == 125000)
#if (RCC_APB1_CLOCK  ==  45000000)
        .Init.Prescaler = 24,       /* CAN_CLOCK = APB1 = 45MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 45MHz/12/14 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_8TQ,
#elif(RCC_APB1_CLOCK  ==  42000000)
        .Init.Prescaler = 24,       /* CAN_CLOCK = APB1 = 42MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_6TQ,    /* --> CAN = 42MHz/12/14 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_7TQ,
#elif RCC_APB1_CLOCK  ==  32000000
        .Init.Prescaler = 32,       /* CAN_CLOCK = APB1 = 32MHz */
                                    /* resulting CAN speed: APB1/prescaler/sumOfTimequants */
                                    /* sum: 1tq for sync + TimeSeg1 + TimeSeg2 */
        .Init.TimeSeg1 = CAN_BS1_5TQ,    /* --> CAN = 32MHz/16/8 = 0.125MHz */
        .Init.TimeSeg2 = CAN_BS2_2TQ,
#else
#error "Please configure CAN Baudrate according to your clock configuration "
#endif
#endif
        .Init.Mode = CAN_MODE_NORMAL,    /* for test purpose, without connected can-bus use LOOPBACK mode */
        .Init.SyncJumpWidth = CAN_SJW_1TQ,
        .Init.TimeTriggeredMode = DISABLE,    /* time triggerd communication mode */
                            /* DISABLE: no influence */
                            /* ENABLE: saves timestamps for received and transmitted messages. See reference manual
                            for more information. */
        .Init.AutoBusOff = ENABLE,    /* automatic bus-off management */
                            /* DISABLE: Manually re-initialize CAN and wait for 128 * 11 recessive bits */
                            /* ENABLE: automatically leave bus-off mode after 128 * 11 recessive bits */
        .Init.AutoWakeUp = ENABLE,     /* automatic wake-up mode */
                            /* ENABLE: automatically leave sleep mode on message receiving */
                            /* DISABLE: SLEEP bit needs to be deleted by software */
        .Init.AutoRetransmission = DISABLE,     /* automatic retransition mode; */
                            /* DISABLE: retransmit the message until it has been successfully transmitted */
                            /* ENABLE: transmit only once, independently of transmission result */
        .Init.ReceiveFifoLocked = ENABLE,    /* Receive FIFO locked against overrun. */
                            /* DISABLE: A new incoming message overwrites the last received message. */
                            /* ENABLE: Once a receive FIFO is full the next incoming message will be discarded. */
        .Init.TransmitFifoPriority = ENABLE,     /* Transmit FIFO priority */
                            /* DISABLE: driven by identifier of message. Lower identifier equals higher priority */
                            /* ENABLE: driven chronologically */
};



/* ***************************************
 *  Configure TX messages here
 ****************************************/

const CAN_MSG_TX_TYPE_s can_CAN0_messages_tx[] = {
        { 0x110, 8, 100, 0, NULL_PTR },  /*!< BMS system state 0 */
        { 0x111, 8, 100, 0, NULL_PTR },  /*!< BMS system state 1 */
        { 0x112, 8, 100, 0, NULL_PTR },  /*!< BMS system state 2 */

        { 0x115, 8, 100, 0, NULL_PTR },  /*!< BMS slave state 0 */
        { 0x116, 8, 100, 0, NULL_PTR },  /*!< BMS slave state 1 */

        { 0x130, 8, 100, 30, NULL_PTR },  /*!< Maximum allowed cur
        { 0x131, 8, 100, 30, NULL_PTR },  /*!< SOP */
        { 0x140, 8, 1000, 30, NULL_PTR },  /*!< SOC */
        { 0x150, 8, 5000, 30, NULL_PTR },  /*!< SOH */
        { 0x160, 8, 1000, 30, NULL_PTR },  /*!< SOE */
        { 0x170, 8, 100, 30, NULL_PTR },  /*!< Cell voltages Min Max Average */
        { 0x171, 8, 100, 30, NULL_PTR },  /*!< SOV */
        { 0x180, 8, 100, 30, NULL_PTR },  /*!< Cell temperatures Min Max Average */
        { 0x190, 8, 1000, 30, NULL_PTR },  /*!< Tempering */
        { 0x1A0, 8, 1000, 30, NULL_PTR },  /*!< Insulation */

        { 0x1D0, 8, 1000, 40, NULL_PTR },  /*!< Running average power 0 */
        { 0x1D1, 8, 1000, 40, NULL_PTR },  /*!< Running average power 1 */
        { 0x1D2, 8, 1000, 40, NULL_PTR },  /*!< Running average power 2 */
        { 0x1E0, 8, 1000, 40, NULL_PTR },  /*!< Running average current 0 */
        { 0x1E1, 8, 1000, 40, NULL_PTR },  /*!< Running average current 1 */
        { 0x1E2, 8, 1000, 40, NULL_PTR },  /*!< Running average current 2 */

        { 0x1F0, 8, 1000, 40, NULL_PTR },  /*!< Pack voltage */
```

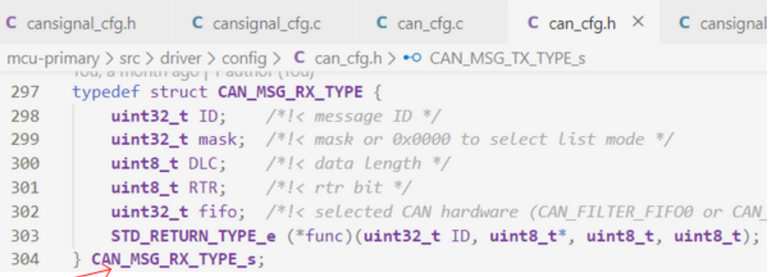
The message ID we need to change

```
        { 0x200, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 0 1 2 */
        { 0x201, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 3 4 5 */
        { 0x202, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 6 7 8 */
        { 0x203, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 9 10 11 */
        { 0x204, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 12 13 14 */
        { 0x205, 8, 200, 20, NULL_PTR },  /*!< Cell voltages module 0 cells 15 16 17 */

        { 0x210, 8, 200, 30, NULL_PTR },  /*!< Cell temperatures module 0 cells 0 1 2 */
        { 0x211, 8, 200, 30, NULL_PTR },  /*!< Cell temperatures module 0 cells 3 4 5 */
        { 0x212, 8, 200, 30, NULL_PTR },  /*!< Cell temperatures module 0 cells 6 7 8 */
        { 0x213, 8, 200, 30, NULL_PTR },  /*!< Cell temperatures module 0 cells 9 10 11 */

        { 0x220, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 0 1 2 */
        { 0x221, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 3 4 5 */
        { 0x222, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 6 7 8 */
        { 0x223, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 9 10 11 */
        { 0x224, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 12 13 14 */
        { 0x225, 8, 200, 40, NULL_PTR },  /*!< Cell voltages module 1 cells 15 16 17 */

        { 0x230, 8, 200, 50, NULL_PTR },  /*!< Cell temperatures module 1 cells 0 1 2 */
        { 0x231, 8, 200, 50, NULL_PTR },  /*!< Cell temperatures module 1 cells 3 4 5 */
        { 0x232, 8, 200, 50, NULL_PTR },  /*!< Cell temperatures module 1 cells 6 7 8 */
```


C cansignal_cfg.h
C cansignal_cfg.c
C can_cfg.c
C can_cfg.h ×
C cans
mcu-primary > src > driver > config > C can_cfg.h > CAN_MSG_TX_TYPE_s

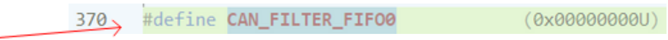
```
typedef struct  {
    uint32_t ID;                    /*!< CAN message id */
    uint8_t DLC;                    /*!< CAN message data length coc
    uint32_t repetition_time;       /*!< CAN message cycle time */
    uint32_t repetition_phase;      /*!< CAN message startup (first
    can_callback_funcPtr cbk_func;  /*!< CAN message callback after
} CAN_MSG_TX_TYPE_s;        You, a month ago • Add all foxBMS files
```

```
        { 0x233, 8, 200, 50, NULL_PTR },  /*!< Cell temperatures module 1 cells 9 10 11 */

        { 0x240, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 0 1 2 */
        { 0x241, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 3 4 5 */
        { 0x242, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 6 7 8 */
        { 0x243, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 9 10 11 */
        { 0x244, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 12 13 14 */
        { 0x245, 8, 200, 60, NULL_PTR },  /*!< Cell voltages module 2 cells 15 16 17 */

        { 0x250, 8, 200, 70, NULL_PTR },  /*!< Cell temperatures module 2 cells 0 1 2 */
        { 0x251, 8, 200, 70, NULL_PTR },  /*!< Cell temperatures module 2 cells 3 4 5 */
        { 0x252, 8, 200, 70, NULL_PTR },  /*!< Cell temperatures module 2 cells 6 7 8 */
        { 0x253, 8, 200, 70, NULL_PTR },  /*!< Cell temperatures module 2 cells 9 10 11 */

        { 0x260, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 0 1 2 */
        { 0x261, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 3 4 5 */
        { 0x262, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 6 7 8 */
        { 0x263, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 9 10 11 */
        { 0x264, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 12 13 14 */
        { 0x265, 8, 200, 80, NULL_PTR },  /*!< Cell voltages module 3 cells 15 16 17 */

        { 0x270, 8, 200, 90, NULL_PTR },  /*!< Cell temperatures module 3 cells 0 1 2 */
        { 0x271, 8, 200, 90, NULL_PTR },  /*!< Cell temperatures module 3 cells 3 4 5 */
        { 0x272, 8, 200, 90, NULL_PTR },  /*!< Cell temperatures module 3 cells 6 7 8 */
        { 0x273, 8, 200, 90, NULL_PTR },  /*!< Cell temperatures module 3 cells 9 10 11 */

        { 0x280, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 0 1 2 */
        { 0x281, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 3 4 5 */
        { 0x282, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 6 7 8 */
        { 0x283, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 9 10 11 */
        { 0x284, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 12 13 14 */
        { 0x285, 8, 200, 100, NULL_PTR },  /*!< Cell voltages module 4 cells 15 16 17 */

        { 0x290, 8, 200, 110, NULL_PTR },  /*!< Cell temperatures module 4 cells 0 1 2 */
        { 0x291, 8, 200, 110, NULL_PTR },  /*!< Cell temperatures module 4 cells 3 4 5 */
        { 0x292, 8, 200, 110, NULL_PTR },  /*!< Cell temperatures module 4 cells 6 7 8 */
        { 0x293, 8, 200, 110, NULL_PTR },  /*!< Cell temperatures module 4 cells 9 10 11 */

        { 0x2A0, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 0 1 2 */
        { 0x2A1, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 3 4 5 */
        { 0x2A2, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 6 7 8 */
        { 0x2A3, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 9 10 11 */
        { 0x2A4, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 12 13 14 */
        { 0x2A5, 8, 200, 120, NULL_PTR },  /*!< Cell voltages module 5 cells 15 16 17 */

        { 0x2B0, 8, 200, 130, NULL_PTR },  /*!< Cell temperatures module 5 cells 0 1 2 */
        { 0x2B1, 8, 200, 130, NULL_PTR },  /*!< Cell temperatures module 5 cells 3 4 5 */
        { 0x2B2, 8, 200, 130, NULL_PTR },  /*!< Cell temperatures module 5 cells 6 7 8 */
        { 0x2B3, 8, 200, 130, NULL_PTR },  /*!< Cell temperatures module 5 cells 9 10 11 */

        { 0x2C0, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 0 1 2 */
        { 0x2C1, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 3 4 5 */
        { 0x2C2, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 6 7 8 */
        { 0x2C3, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 9 10 11 */
        { 0x2C4, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 12 13 14 */
        { 0x2C5, 8, 200, 140, NULL_PTR },  /*!< Cell voltages module 6 cells 15 16 17 */

        { 0x2D0, 8, 200, 150, NULL_PTR },  /*!< Cell temperatures module 6 cells 0 1 2 */
        { 0x2D1, 8, 200, 150, NULL_PTR },  /*!< Cell temperatures module 6 cells 3 4 5 */
        { 0x2D2, 8, 200, 150, NULL_PTR },  /*!< Cell temperatures module 6 cells 6 7 8 */
        { 0x2D3, 8, 200, 150, NULL_PTR },  /*!< Cell temperatures module 6 cells 9 10 11 */

        { 0x2E0, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 0 1 2 */
        { 0x2E1, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 3 4 5 */
        { 0x2E2, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 6 7 8 */
        { 0x2E3, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 9 10 11 */
        { 0x2E4, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 12 13 14 */
        { 0x2E5, 8, 200, 160, NULL_PTR },  /*!< Cell voltages module 7 cells 15 16 17 */

        { 0x2F0, 8, 200, 170, NULL_PTR },  /*!< Cell temperatures module 7 cells 0 1 2 */
        { 0x2F1, 8, 200, 170, NULL_PTR },  /*!< Cell temperatures module 7 cells 3 4 5 */
        { 0x2F2, 8, 200, 170, NULL_PTR },  /*!< Cell temperatures module 7 cells 6 7 8 */
        { 0x2F3, 8, 200, 170, NULL_PTR },  /*!< Cell temperatures module 7 cells 9 10 11 */

#ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
        , { 0x35B, 8, 100, 20, NULL_PTR }  /*!< Current Sensor Trigger *
#endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
};


const CAN_MSG_TX_TYPE_s can_CAN1_messages_tx[] = {
};

const uint8_t can_CAN0_tx_length = sizeof(can_CAN0_messages_tx)/sizeof(can_CAN0_messages_tx[0]);
const uint8_t can_CAN1_tx_length = sizeof(can_CAN1_messages_tx)/sizeof(can_CAN1_messages_tx[0]);
```


C cansignal_cfg.h
C cansignal_cfg.c
C can_cfg.c
C can_cfg.h ×
C cansignal
mcu-primary > src > driver > config > C can_cfg.h > CAN_MSG_TX_TYPE_s

```
typedef struct CAN_MSG_RX_TYPE {
    uint32_t ID;    /*!< message ID */
    uint32_t mask;  /*!< mask or 0x0000 to select list mode */
    uint8_t DLC;    /*!< data length */
    uint8_t RTR;    /*!< rtr bit */
    uint32_t fifo;  /*!< selected CAN hardware (CAN_FILTER_FIFO0 or CAN_
    STD_RETURN_TYPE_e (*func)(uint32_t ID, uint8_t*, uint8_t, uint8_t);
} CAN_MSG_RX_TYPE_s;
```

```
/* ***************************************
 *  Configure RX messages here
 ****************************************/

/* Bypassed messages are --- ALSO --- to be configured here. See further down for bypass ID setting!  */
CAN_MSG_RX_TYPE_s can0_RxMsgs[] = {
        { 0x120, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< state request      */

        { CAN_ID_SOFTWARE_RESET_MSG, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< software reset     */
```



```
#define CAN_FILTER_FIFO0            (0x00000000U)
```

```
#ifdef CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED
        { 0x35C, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< current sensor I   */
        { 0x35D, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< current sensor U1  */
        { 0x35E, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< current sensor U2  */
        { 0x35F, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },   /*!< current sensor U3  */
        { 0x525, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor T in cyclic mode  */
        { 0x526, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor Power in cyclic mode  */
        { 0x527, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor C-C in cyclic mode  */
```

```c
        { 0x528, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor E-C in cyclic mode  */
#else /* CURRENT_SENSOR_ISABELLENHUETTE_CYCLIC */
        { 0x521, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor I in cyclic mode   */
        { 0x522, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor U1 in cyclic mode  */
        { 0x523, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor U2 in cyclic mode  */
        { 0x524, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor U3 in cyclic mode  */
        { 0x525, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor T in cyclic mode  */
        { 0x526, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor Power in cyclic mode  */
        { 0x527, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor C-C in cyclic mode  */
        { 0x528, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< current sensor E-C in cyclic mode  */
#endif /* CURRENT_SENSOR_ISABELLENHUETTE_TRIGGERED */
        { 0x100, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< debug message      */
        { 0x777, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< request SW version */
        { 0x121, 0xFFFF, 8, 0, CAN_FILTER_FIFO0, NULL },    /*!< engine request */
};


CAN_MSG_RX_TYPE_s can1_RxMsgs[] = {
};


const uint8_t can_CAN0_rx_length = sizeof(can0_RxMsgs)/sizeof(can0_RxMsgs[0]);
const uint8_t can_CAN1_rx_length = sizeof(can1_RxMsgs)/sizeof(can1_RxMsgs[0]);

/* ***************************************
 *  Set bypass message IDs here
 ****************************************/

/* These IDs have to be included in the configuration for the filters in can_RxMsgs[]! */
uint32_t can0_bufferBypass_RxMsgs[CAN0_BUFFER_BYPASS_NUMBER_OF_IDs] = { CAN_ID_SOFTWARE_RESET_MSG };

uint32_t can1_bufferBypass_RxMsgs[CAN1_BUFFER_BYPASS_NUMBER_OF_IDs] = {};

```