```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    cansignal.h
 * @author  foxBMS Team
 * @date    01.10.2015 (date of creation)
 * @ingroup DRIVERS
 * @prefix  CANS
 *
 * @brief   Headers for the messages and signal settings for the CAN driver
 *
 * generic conversion module header of Can signals from CAN buffered reception to
 * DATA Manager and vice versa
 *
 */

#ifndef CANSIGNAL_H_
#define CANSIGNAL_H_

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

#include "can.h"

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

/*================== Constant and Variable Definitions ====================*/
/**
 * This structure contains variables relevant for the CAN signal module.
 */
typedef struct {
```

Booleans should have been used.

```c
    uint8_t periodic_enable;                    /*!< defines if periodic transmit and receive should run  */
    uint8_t current_sensor_present;             /*!< defines if a current sensor is detected  */
    uint8_t current_sensor_cc_present;          /*!< defines if a CC info is being sent  */
} CANS_STATE_s;
```

CC = Coulomb Count?

This should have been more specific to
convey the needed info.

```c
/*================== Function Prototypes ==================================*/
/**
 * initializes local variables and module internals needed to use conversion of
 * can signals. Until now no initialization is needed and thus the function does
 * nothing.
 */
extern void CANS_Init(void);

/**
 * handles the conversion of can signals from and to datamanager database or
 * other modules defined by the getter and setter configuration.
 */
extern void CANS_MainFunction(void);

extern void CANS_Enable_Periodic(uint8_t command);
extern uint8_t CANS_IsCurrentSensorPresent(void);
extern uint8_t CANS_IsCurrentSensorCCPresent(void);

/**
 * @brief  Add message to transmit buffer, message will be transmitted shortly after.
 *
 * @param  canNode: canNode on which the message shall be transmitted
 * @param  msgID:    ID of the message that will be transmitted
 * @param  ptrMsgData:    pointer to a uint8_t array that contains the message that will be transmitted
 * @param  msgLength:    length of the message that will be transmitted
 *                  This parameter can be a value of CAN_identifier_type.
 * @param  RTR     Specifies the type of frame for the message that will be transmitted.
 *                  This parameter can be a value of CAN_remote_transmission_request
 *
 * @retval E_OK if successful, E_NOT_OK if buffer is full or error occurred
 */                             This is a function called by CANS_PeriodicTransmit. How can this be a public function?
extern STD_RETURN_TYPE_e CANS_AddMessage(CAN_NodeTypeDef_e canNode, uint32_t msgID, uint8_t* ptrMsgData,
        uint32_t msgLength, uint32_t RTR);

/**
 * @brief  Transmits canNode transmit buffer
 *
 * @param canNode:  canNode on which the message shall be transmitted
 *
 * @retval E_OK if transmission successful, otherwise E_NOT_OK
 */
extern STD_RETURN_TYPE_e CANS_TransmitBuffer(CAN_NodeTypeDef_e canNode);


/**
 * @brief  Transmits message directly on the CAN bus
 *
 * @param  canNode: canNode on which the message shall be transmitted
 * @param  msgID:    ID of the message that will be transmitted
 * @param  ptrMsgData:    pointer to the data that shall be transmitted
 * @param  msgLength:    Specifies the data length
 * @param  RTR: Specifies the type of frame for the message that will be transmitted.
 *
 * @retval E_OK if transmission successful, otherwise E_NOT_OK
 */
extern STD_RETURN_TYPE_e CANS_TransmitMessage(CAN_NodeTypeDef_e canNode, uint32_t msgID, uint8_t* ptrMsgData,
        uint32_t msgLength, uint32_t RTR);

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

#endif /* CANSIGNAL_H_ */
```