```
/**
 *
 * @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    database.h
 * @author  foxBMS Team
 * @date    18.08.2015 (date of creation)
 * @ingroup ENGINE
 * @prefix  DATA
 *
 * @brief   Database module header
 *
 * Provides interfaces to database module
 *
```

```c
 */

#ifndef DATABASE_H_
#define DATABASE_H_

/*================== Includes =============================================*/
#include "database_cfg.h"
#include "os.h"

/*================== Macros and Definitions ==================================*/
/**
 * struct for database queue, contains pointer to data, database entry and access type
 */
typedef struct {
    /* FIXME what is the intention of this union? isn't it dangerous if someone expects a pointer to and accesses via .u32ptr, but there is a value stored in value? */
    union {
        uint32_t                u32value;    /*  reference by uint32_t value   */
        uint32_t                *u32ptr;    /*  reference by uint32_t pointer */
        void                    *voidptr;    /*  reference by general pointer */
    } value;
    DATA_BLOCK_ID_TYPE_e        blockID;    /* definition of used message data type */
    DATA_BLOCK_ACCESS_TYPE_e    accesstype; /* read or write access type */
} DATA_QUEUE_MESSAGE_s;


/**
 * database access type definition: read or write
 */
typedef struct {
    void *RDptr;
    void *WRptr;
} DATA_BLOCK_ACCESS_s;

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

/*================== Extern Constant and Variable Definitions ===============*/
extern QueueHandle_t data_queue;

/*================== Extern Function Prototypes ==============================*/
/**
 * @brief   Initialization of database manager
 */
extern void DATA_Init(void);

/**
 * @brief   Stores a datablock in database
 *
 * Do not call this function from inside a critical section, as it is
 * computationally complex.
 * @param   blockID (type: DATA_BLOCK_ID_TYPE_e)
 * @param   dataptrfromSender (type: void *)
```

Need to change the order of these two lines to align with the parameters of the function.

```c
 */
extern void DB_WriteBlock(void *dataptrfromSender, DATA_BLOCK_ID_TYPE_e  blockID);

/**
 * @brief   Reads a datablock in database by value.
 *
 * Do not call this function from inside a critical section, as it is
 * computationally complex.
 * @param   blockID (type: DATA_BLOCK_ID_TYPE_e)
 * @param   dataptrtoReceiver (type: void *)
 * @return  STD_RETURN_TYPE_e
 */
extern STD_RETURN_TYPE_e DB_ReadBlock(void *dataptrtoReceiver, DATA_BLOCK_ID_TYPE_e  blockID);

 /**
  * @brief   trigger of database manager
  */
extern void DATA_Task(void);

#endif /* DATABASE_H_ */

```