```c
/******************************************************************************
 *
 *    ________  _________  ____    ____  _____          ________   ______
 *   |_   __  ||  _   _  ||_   \  /   _||_   _|         |_   __  |.' ____ \
 *     | |_ \_||_/ | | \_|  |   \/   |    | |     ______ | |_ \_|| (___ \_|
 *     |  _| _     | |      | |\  /| |    | |   _|______||  _| _   _.____`.
 *    _| |__/ |   _| |_    _| |_\/_| |_  _| |__/ |      _| |__/ || \____) |
 *   |________|  |_____|  |_____||_____||________|     |________| \______.'
 *
 ******************************************************************************
 *
 * File           : fifo.h
 * Version        : 1.0
 *
 ******************************************************************************
 *
 * Description    : Managing a FIFO using descriptor and pointers
 *                  Maximal size of FIFO is 255
 *
 ******************************************************************************
 *
 * Author         : Miguel Santos
 * Date           : 14.09.2023
 *
 ******************************************************************************
 *
 * MPLAB X        : 5.45
 * XC32           : 2.50
 * Harmony        : 2.06
 *
 *****************************************************************************/

#ifndef FIFO_H
#define FIFO_H

/******************************************************************************/

#include <stdint.h>
#include <stdbool.h>

/******************************************************************************/

/* FIFO descriptor structure */
typedef struct fifo {
   uint16_t size;
   uint8_t *write;
   uint8_t *read;
   uint8_t *start;
   uint8_t *end;
} S_Fifo;

/******************************************************************************/

/**
 * @brief FIFO_Init
 *
 * This function initializes a FIFO with the provided parameters,
 * setting its size, start address, and initializing all elements
 * to the given initial value.
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @param fifoSize       The size of the FIFO.
 * @param fifoStart      Pointer to the beginning of the FIFO memory.
 * @param initialValue   The initial value to set for all elements in the FIFO.
 */
void FIFO_Initialize( S_Fifo *fifoDescriptor, uint16_t fifoSize,
                 uint8_t *fifoStart, uint8_t initialValue );

/******************************************************************************/

/**
 * @brief FIFO_GetWriteSpace
 *
 * This function calculates the available space for writing
```

```
 * in the provided FIFO descriptor,
 * taking into account the current read and write positions.
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @return The available space for writing in the FIFO.
 */
uint8_t FIFO_GetWriteSpace( S_Fifo *fifoDescriptor );

/*****************************************************************************/

/**
 * @brief FIFO_GetReadSpace
 *
 *
 * This function calculates the available space for reading
 * from the provided FIFO descriptor,
 * taking into account the current read and write positions.
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @return The available space for reading from the FIFO.
 */
uint8_t FIFO_GetReadSpace( S_Fifo *fifoDescriptor );

/*****************************************************************************/

/**
 * @brief FIFO_Add
 *
 * This function attempts to put the specified character into the FIFO.
 * If the FIFO is full, returns 0 (FIFO FULL),
 * otherwise, it puts the character and returns 1 (OK).
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @param value          The value to add to the FIFO.
 * @return 1 if (OK), 0 if (FIFO FULL).
 */
bool FIFO_Add( S_Fifo *fifoDescriptor , uint8_t value );

/*****************************************************************************/

/**
 * @brief FIFO_GetData
 *
 * This function attempts to get a value from the FIFO.
 * If the FIFO is empty, returns 0 (FIFO EMPTY),
 * otherwise, it gets the value and returns 1 (OK).
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @param value          Pointer to store the retrieved value.
 * @return 1 if (OK), 0 if (FIFO EMPTY).
 */
bool FIFO_GetData( S_Fifo *fifoDescriptor , uint8_t *value );

/*****************************************************************************/

/**
 * @brief FIFO_GetBuffer
 *
 * This function attempts to get all the FIFO in a buffer.
 * If the FIFO is empty, returns 0 (FIFO EMPTY),
 * otherwise, it gets the value and returns 1 (OK).
 *
 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
 * @param buffer         Pointer to the buffer to sore the FIFO.
 * @return true if (OK), false if (FIFO EMPTY).
 */
bool FIFO_GetBuffer( S_Fifo *fifoDescriptor , uint8_t *buffer );

/*****************************************************************************/

#endif

/* End of File ***************************************************************/
```