```c
/*******************************************************************************
 *
 *    ________  _________  ____    ____  _____          ________   ______
 *   |_   __  ||  _   _  ||_   \  /   _||_   _|        |_   __  |.' ____ \
 *     | |_ \_||_/ | | \_|  |   \/   |    | |     ______ | |_ \_|| (___ \_|
 *     |  _| _     | |      | |\  /| |    | |   _|______||  _| _  _.____`.
 *    _| |__/ |   _| |_    _| |_\/_| |_  _| |__/ |      _| |__/ || \____) |
 *   |________|  |_____|  |_____||_____||________|     |________| \______.'
 *
 *******************************************************************************
 *
 * File          : fifo.c
 * Version       : 1.0
 *
 *******************************************************************************
 *
 * Description   : Managing a FIFO using descriptor and pointers
 *                 Maximal size of FIFO is 255
 *
 *******************************************************************************
 *
 * Author        : Miguel Santos
 * Date          : 25.09.2023
 *
 *******************************************************************************
 *
 * MPLAB X       : 5.45
 * XC32          : 2.50
 * Harmony       : 2.06
 *
 ******************************************************************************/

#include "FIFO.h"

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

/**
 * @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.
 * @return void
 */
void FIFO_Initialize( S_Fifo *fifoDescriptor, uint16_t fifoSize,
                uint8_t *fifoStart, uint8_t initialValue )
{
    /* Local variables declaration */
    uint8_t *fifoPosition;
    uint16_t i;

    fifoPosition = fifoStart;

    /* Fifo descriptor values initialisation */
    fifoDescriptor->size    = fifoSize;
    fifoDescriptor->start   = fifoStart;
    fifoDescriptor->end     = fifoStart + fifoSize - 1;
    fifoDescriptor->write   = fifoStart;
    fifoDescriptor->read    = fifoStart;

    /* Loop through entire fifo to set initial value */
    for( i = 0 ; i < fifoSize ; i++)
    {
        fifoPosition[i] = initialValue;
    }
}

/******************************************************************************/
```

```
74  /**
75   * @brief FIFO_GetWriteSpace
76   *
77   * This function calculates the available space for writing
78   * in the provided FIFO descriptor,
79   * taking into account the current read and write positions.
80   *
81   * @param fifoDescriptor Pointer to the FIFO descriptor structure.
82   * @return The available space for writing in the FIFO.
83   */
84  uint8_t FIFO_GetWriteSpace( S_Fifo *fifoDescriptor )
85  {
86      /* Local variables declaration */
87      int32_t writeSpace;
88
89      /* Calculate space available */
90      writeSpace = fifoDescriptor->read - fifoDescriptor->write - 1;
91
92      /* Adjust to positive if needed */
93      if (writeSpace < 0)
94      {
95          writeSpace = writeSpace + fifoDescriptor->size;
96      }
97
98      /* Return value */
99      return (uint16_t)writeSpace;
100 }
101
102 /*****************************************************************************/
103
104 /**
105  * @brief FIFO_GetReadSpace
106  *
107  *
108  * This function calculates the available space for reading
109  * from the provided FIFO descriptor,
110  * taking into account the current read and write positions.
111  *
112  * @param fifoDescriptor Pointer to the FIFO descriptor structure.
113  * @return The available space for reading from the FIFO.
114  */
115 uint8_t FIFO_GetReadSpace( S_Fifo *fifoDescriptor )
116 {
117     /* Local variables declaration */
118     int32_t readSpace;
119
120     /* Calculate space available */
121     readSpace = fifoDescriptor->write - fifoDescriptor->read;
122
123     /* Adjust to positive if needed */
124     if (readSpace < 0)
125     {
126         readSpace = readSpace +  fifoDescriptor->size;
127     }
128
129     /* Return value */
130     return (uint16_t)readSpace;
131 }
132
133 /*****************************************************************************/
134
135 /**
136  * @brief FIFO_Add
137  *
138  * This function attempts to put the specified character into the FIFO.
139  * If the FIFO is full, returns 0 (FIFO FULL),
140  * otherwise, it puts the character and returns 1 (OK).
141  *
142  * @param fifoDescriptor Pointer to the FIFO descriptor structure.
143  * @param value          The value to add to the FIFO.
144  * @return true if (OK), false if (FIFO FULL).
145  */
146 bool FIFO_Add( S_Fifo *fifoDescriptor , uint8_t value )
```

```
147 {
148     /* Local variables declaration */
149     bool writeStatus;
150 
151     /* True = space available ; False = FIFO full */
152     writeStatus = FIFO_GetWriteSpace(fifoDescriptor);
153 
154     if (writeStatus)
155     {
156         /* Write the value into the FIFO */
157         *(fifoDescriptor->write) = value;
158 
159         /* Increment the write pointer */
160         fifoDescriptor->write++;
161 
162         /* Handle wrap-around */
163         if (fifoDescriptor->write > fifoDescriptor->end)
164         {
165             fifoDescriptor->write = fifoDescriptor->start;
166         }
167     }
168 
169     /* Return status */
170     return writeStatus;
171 }
172 
173 /*****************************************************************************/
174 
175 /**
176  * @brief FIFO_GetData
177  *
178  * This function attempts to get a value from the FIFO.
179  * If the FIFO is empty, returns 0 (FIFO EMPTY),
180  * otherwise, it gets the value and returns 1 (OK).
181  *
182  * @param fifoDescriptor Pointer to the FIFO descriptor structure.
183  * @param value          Pointer to store the retrieved value.
184  * @return true if (OK), false if (FIFO EMPTY).
185  */
186 bool FIFO_GetData( S_Fifo *fifoDescriptor , uint8_t *value )
187 {
188     /* Local variables declaration */
189     bool readStatus;
190 
191     /* True = values in FIFO ; False = FIFO empty */
192     readStatus = FIFO_GetReadSpace(fifoDescriptor);
193 
194     if (readStatus)
195     {
196         /* Read value in FIFO */
197         *value = *(fifoDescriptor->read);
198 
199         /* Increment read pointer */
200         fifoDescriptor->read++;
201 
202         /* Handle wrap-around */
203         if (fifoDescriptor->read > fifoDescriptor->end)
204         {
205             fifoDescriptor->read = fifoDescriptor->start;
206         }
207     }
208     else
209     {
210         /* Value read = NULL */
211         *value = 0;
212     }
213 
214     /* Return status */
215     return readStatus;
216 }
217 
218 /*****************************************************************************/
219 
```

```
220  /**
221   * @brief FIFO_GetBuffer
222   *
223   * This function attempts to get all the FIFO in a buffer.
224   * If the FIFO is empty, returns 0 (FIFO EMPTY),
225   * otherwise, it gets the value and returns 1 (OK).
226   *
227   * @param fifoDescriptor Pointer to the FIFO descriptor structure.
228   * @param buffer         Pointer to the buffer to sore the FIFO.
229   * @return true if (OK), false if (FIFO EMPTY).
230   */
231  bool FIFO_GetBuffer( S_Fifo *fifoDescriptor , uint8_t *buffer )
232  {
233      /* Local variables declaration */
234      bool readStatus;
235      uint8_t value;
236      uint8_t *p_buffer;
237
238      readStatus = false;
239      value = 0x00;
240      p_buffer = buffer;
241
242      /* True = values in FIFO ; False = FIFO empty */
243      while(FIFO_GetData(fifoDescriptor, &value))
244      {
245          *p_buffer = value;
246          p_buffer++;
247          readStatus = true;
248      }
249
250      /* Return status */
251      return readStatus;
252  }
253
254  /*****************************************************************************/
255
256  /* End of File ***************************************************************/
257
```