```
 1 /**
 2  * @file
 3  * File:                message.h
 4  * Author:              Manuel Federanko
 5  * Version:             1.0
 6  * Comments:
 7  * Revision history:
 8  */
 9
10 #ifndef __MESSAGE_H
11 #define __MESSAGE_H
12
13 #include <xc.h>
14 #include "types.h"
15
16 #ifdef  __cplusplus
17 extern "C" {
18 #endif
19
20 #define NONE                    '0'
21 #define STRING                  '1'
22 #define INT8                    '2'
23 #define INT16                   '3'
24 #define INT24                   '4'
25 #define INT32                   '5'
26 #define UINT8                   '6'
27 #define UINT16                  '7'
28 #define UINT24                  '8'
29 #define UINT32                  '9'
30 #define FLOAT                   'F'
31
32 #define START_OF_TEXT             2
33 #define END_OF_TEXT               3
34 #define RECEIVEBUFF_SIZE          8
35
36         char receive_buff[RECEIVEBUFF_SIZE];
37         int8_t buffpos;
38
39         /**
40          * Initialises the messaging buffer and pointers.
41          */
42         void initMessaging();
43
44         /**
45          * Counts the available Characters in the receive buffer and returns
46          * that number.
47          * @return the number of available characters
48          */
49         int8_t charAvailable();
50
51         /**
52          * Reads the next Character and advances the buffer by one.
53          * @return the read character
54          */
55         char readNext();
56
```

```
 57          /**
 58           * Reads a package of data and returns a value unequal to 0 if the packag
 59           * is finished.
 60           * @return 0 if the package is not finshied, otherwise 1
 61           */
 62          char readPackage();
 63
 64          /**
 65           * Returns 1 if at least one byte has been written to the buffer,
 66           * otherwise 0.
 67           * @return 1 if a byte is in the buffer, otherwise 0
 68           */
 69          char packageStarted();
 70
 71          /**
 72           * The return value behaves in the same way as does readPackage().
 73           * @return 0 if the package is not finshied, otherwise 1
 74           */
 75          char packageFinished();
 76
 77          /**
 78           * Returns the type of the package as a Character.
 79           * @return the type
 80           */
 81          char getType();
 82
 83          /**
 84           * Sets the position of the buffer to the beginning of the buffer
 85           * effectively resetting the buffer.
 86           */
 87          void seekFront();
 88
 89          /**
 90           * Clears the whole buffer with 0s (zeros).
 91           */
 92          void clear();
 93
 94          /**
 95           * Reads an 8 bit signed Integer from the buffer.
 96           * @return the 8 bit Integer
 97           */
 98          int8_t readInt8();
 99          /**
100           * Reads an 16 bit signed Integer from the buffer.
101           * @return the 16 bit Integer
102           */
103          int16_t readInt16();
104          /**
105           * Reads an 24 bit signed Integer from the buffer.
106           * @return the 24 bit Integer
107           */
108          int24_t readInt24();
109          /**
110           * Reads an 32 bit signed Integer from the buffer.
111           * @return the 32 bit Integer
112           */
113          int32_t readInt32();
```

```
114         /**
115          * Reads an 3 byte Float from the buffer.
116          * @return the Float
117          */
118         float readFloat();
119 
120         /**
121          * Reads a string from the buffer into the specified char array.
122          * @param c the destination
123          */
124         void readString(char **c);
125 
126         /**
127          * Sends an 8 bit wide Integer variable over the USART register
128          * @param i the data
129          */
130         void sendInt8(int8_t i);
131         /**
132          * Sends an 16 bit wide Integer variable over the USART register
133          * @param i the data
134          */
135         void sendInt16(int16_t i);
136         /**
137          * Sends an 24 bit wide Integer variable over the USART register
138          * @param i the data
139          */
140         void sendInt24(int24_t i);
141         /**
142          * Sends an 32 bit wide Integer variable over the USART register
143          * @param i the data
144          */
145         void sendInt32(int32_t i);
146 
147         /**
148          * Sends an 8 bit wide unsigned Integer variable over the USART register
149          * @param i the data
150          */
151         void sendUInt8(uint8_t i);
152 
153         /**
154          * Sends an 16 bit wide unsigned Integer variable over the USART register
155          * @param i the data
156          */
157         void sendUInt16(uint16_t i);
158 
159         /**
160          * Sends an 24 bit wide unsigned Integer variable over the USART register
161          * @param i the data
162          */
163         void sendUInt24(uint24_t i);
164 
165         /**
166          * Sends an 32 bit wide unsigned Integer variable over the USART register
167          * @param i the data
168          */
169         void sendUint32(uint32_t i);
170 
```

```
171         /**
172          * Sends a 3 byte wide Float over the USART module.
173          * @param f the Float to send
174          */
175         void sendFloat(float f);
176 
177         /**
178          * Sends a null-terminated String over the USART module.
179          * @param val the pointer to the first element of the String
180          */
181         void sendString(char *val);
182 
183 #ifdef  __cplusplus
184 }
185 #endif
186 
187 #endif
```