

```

1  /* ***** */
2  /** Descriptive File Name
3
4      @Company
5          ETML-ES
6
7      @File Name
8          mc32_serComm.c
9
10     */
11  /* ***** */
12
13  /* ***** */
14  /* ***** */
15  /* Section: Included Files */
16  /* ***** */
17  /* ***** */
18  #include "Mc32_serComm.h"
19  #include <stdio.h>
20
21  /* This section lists the other files that are included in this file.
22     */
23
24  /* TODO: Include other files here if needed. */
25
26
27  /* ***** */
28  /* ***** */
29  /* Section: File Scope or Global Data */
30  /* ***** */
31  /* ***** */
32
33  /* A brief description of a section can be given directly below the section
34     banner.
35     */
36
37  /* ***** */
38  /** Descriptive Data Item Name
39
40      @Summary
41          Brief one-line summary of the data item.
42
43      @Description
44          Full description, explaining the purpose and usage of data item.
45          <p>
46          Additional description in consecutive paragraphs separated by HTML
47          paragraph breaks, as necessary.
48          <p>
49          Type "JavaDoc" in the "How Do I?" IDE toolbar for more information on tags.
50
51      @Remarks
52          Any additional remarks
53     */
54
55
56  /* ***** */
57  /* ***** */
58  // Section: Local Functions */
59  /* ***** */
60  /* ***** */
61
62
63
64  /* ***** */
65  /* ***** */
66  // Section: Interface Functions */
67  /* ***** */
68  /* ***** */
69
70  /* A brief description of a section can be given directly below the section
71     banner.
72     */
73

```

```

74 // *****
75
76
77 void serDisplayValues ( s_bno055_data *bno055_data )
78 {
79     char sendBuffer[66] = {0};
80     uint8_t i = 0;
81     static uint32_t ctnTimeout = 0;
82
83     /* Preapare Gravity string */
84     sprintf(sendBuffer, "DT: %d0 ms\tGravity : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", (bno055_data->d_time), bno055_data->gravity.x, bno055_data->gravity.y, bno055_data->gravity.z);
85     /* Transmit Gravity string */
86     do{
87         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
88         {
89             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
90             i++;
91         }
92         ctnTimeout++;
93     }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
94     i = 0;
95
96     /* Preapare gyroscope string */
97     sprintf(sendBuffer, "Gyro : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->gyro.x, bno055_data->gyro.y, bno055_data->gyro.z);
98     /* Transmit Gravity string */
99     do{
100         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
101         {
102             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
103             i++;
104         }
105         ctnTimeout++;
106     }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
107     i = 0;
108
109     /* Preapare magnitude string */
110     sprintf(sendBuffer, "Mag : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->mag.x, bno055_data->mag.y, bno055_data->mag.z);
111     /* Transmit Gravity string */
112     do{
113         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
114         {
115             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
116             i++;
117         }
118         ctnTimeout++;
119     }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
120     i = 0;
121
122     /* Preapare linear acceleration string */
123     sprintf(sendBuffer, "Accel : X = %04.03lf\tY = %04.03lf\tZ = %04.03lf \n\r", bno055_data->linear_accel.x, bno055_data->linear_accel.y, bno055_data->linear_accel.z);
124     /* Transmit Gravity string */
125     do{
126         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
127         {
128             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
129             i++;
130         }
131         ctnTimeout++;
132     }while((sendBuffer[i-1] != '\r')&&(ctnTimeout<TIME_OUT));
133     i = 0;
134
135     /* Preapare euler string */
136     sprintf(sendBuffer, "Euler : H = %04.03lf\tP = %04.03lf\tR = %04.03lf \n\r", bno055_data->euler.h, bno055_data->euler.p, bno055_data->euler.r);
137     /* Transmit Gravity string */
138     do{
139         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))

```

```

140         {
141             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
142             i++;
143         }
144         ctnTimeout++;
145     }while((sendBuffer[i-1] != '\r') && (ctnTimeout < TIME_OUT));
146     i = 0;
147
148     /* Preapare quaternion string */
149     sprintf(sendBuffer, "Quater. : W = %05d\tX = %05d\tY = %05d\tZ = %05d \n\n\r",
150             bno055_data->quaternion.w, bno055_data->quaternion.x, bno055_data->quaternion.y,
151             bno055_data->quaternion.z);
152     /* Transmit Gravity string */
153     do{
154         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1))
155         {
156             PLIB_USART_TransmitterByteSend(USART_ID_1, sendBuffer[i]);
157             i++;
158         }
159         ctnTimeout++;
160     }while((sendBuffer[i-1] != '\r') && (ctnTimeout < TIME_OUT));
161     i = 0;
162 }
163
164 void serTransmitString ( USART_MODULE_ID usartId, const char * msg )
165 {
166     char bufferMsg[60] = {0};
167     static uint32_t i = 0;
168     static uint32_t ctnTimeout = 0;
169
170     strncpy(bufferMsg, msg, strlen(msg));
171
172     /* Transmit string */
173     do{
174         if(!PLIB_USART_TransmitterBufferIsFull(usartId))
175         {
176             PLIB_USART_TransmitterByteSend(usartId, bufferMsg[i]);
177             i++;
178         }
179         ctnTimeout++;
180     }while((bufferMsg[i-1] != '\0') && (ctnTimeout < TIME_OUT));
181     i = 0;
182 }
183
184 void serTransmitbuffer ( USART_MODULE_ID usartId, char msg[], uint32_t lenght )
185 {
186     uint32_t i = 0;
187     uint32_t ctnTimeout = 0;
188
189     /* Transmit string */
190     do{
191         if(!PLIB_USART_TransmitterBufferIsFull(usartId))
192         {
193             PLIB_USART_TransmitterByteSend(usartId, msg[i]);
194             i++;
195         }
196         ctnTimeout++;
197     }while((i < lenght) && (ctnTimeout < TIME_OUT));
198     i = 0;
199 }
200
201 bool pollSerialSingleCmd(USART_MODULE_ID usartID, const char * command1)
202 {
203     static char charRead[30] = {0};
204     static uint32_t readCnt = 0;
205
206     // Get command's characters
207     while((PLIB_USART_ReceiverDataIsAvailable(usartID) && (readCnt < 30))){
208         charRead[readCnt] = PLIB_USART_ReceiverByteReceive(usartID);
209         readCnt++;
210     }
211     // Command

```

```

211     if(readCnt >= 30)
212     {
213         /* Reset read counter */
214         readCnt = 0;
215         /* Clear read buffer */
216         memset(charRead,0,strlen(charRead));
217     }
218     // Check occurrence with commands
219     if(strstr(charRead, command1) != NULL) {
220         /* Reset read counter */
221         readCnt = 0;
222         /* Clear read buffer */
223         memset(charRead,0,strlen(charRead));
224         /* Command detected */
225         return true;
226     }
227     else{
228         return false;
229     }
230 }
231
232 bool pollSerialCmds(USART_MODULE_ID usartID, const char * command1, const char *
command2, const char * command3,
233                    const char * command4)
234 {
235     static char charRead[CHAR_READ_BUFFER_SIZE] = {0};
236     static uint32_t readCnt = 0;
237
238     // Get command's characters
239     while((PLIB_USART_ReceiverDataIsAvailable(usartID)&&(readCnt <
CHAR_READ_BUFFER_SIZE)){
240         charRead[readCnt] = PLIB_USART_ReceiverByteReceive(usartID);
241         readCnt++;
242     }
243     // Command
244     if(readCnt >= CHAR_READ_BUFFER_SIZE)
245     {
246         /* Reset read counter */
247         readCnt = 0;
248         /* Clear read buffer */
249         memset(charRead,0,CHAR_READ_BUFFER_SIZE);
250     }
251     // Check occurrence with commands
252     if((strstr(charRead, command1) != NULL) || (strstr(charRead, command2) != NULL)
253        || (strstr(charRead, command3) != NULL) || (strstr(charRead, command4) != NULL
254        )) {
255         /* Reset read counter */
256         readCnt = 0;
257         /* Clear read buffer */
258         memset(charRead,0,CHAR_READ_BUFFER_SIZE);
259         /* Command detected */
260         return true;
261     }
262     else{
263         return false;
264     }
265 }
266 /* *****
267 End of File
268 */

```