```
/*
SD_SS.ino  v 0.921/20160609 - b-gsmgnss 2.105  microSD read/write/delete example utility
COPYRIGHT (c) 2014-2017 Dragos Iosub / R&D Software Solutions srl

*****************************IMPORTANT NOTICE*****************************************
"bgsmgnss_basic_lbr.h", "bgsmgnss_FS_lbr.h" and "bgsmgnss_basic_lbr.h", "bgsmgnss_FS_lbr.ino"
ARE REQUIERED IN ORDER TO RUN THIS EXAMPLE!!!!!!!!!!!!!!!!!!!

Download the "a-gsmII kickstart for Arduino"/"b-gsmgnss kickstart for Arduino" from here:
https://itbrainpower.net/downloads
Uncompress the archive and copy the files mentined above in the folder
where is this utility, then you can compile this code.

You may want to modify the message variable found at line 53
*****************************END of NOTICE*****************************************

You are legaly entitled to use this SOFTWARE ONLY IN CONJUNCTION WITH b-gsmgnss DEVICES USAGE. Modifications, derivates and redistribution
of this software must include unmodified this COPYRIGHT NOTICE. You can redistribute this SOFTWARE and/or modify it under the terms
of this COPYRIGHT NOTICE. Any other usage may be permited only after written notice of Dragos Iosub / R&D Software Solutions srl.

This SOFTWARE is distributed is provide "AS IS" in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Dragos Iosub, Bucharest 2017.
http://itbrainpower.net
*/
/*
In order to make your Arduino serial communication reliable (especially for Arduino Uno) with b-gsmgnss shield, you must edit:
C:\Program Files\Arduino\libraries\SoftwareSerial\SoftwareSerial.h

comment the line 42
#define _SS_MAX_RX_BUFF 64

this, will look after that like:
//#define _SS_MAX_RX_BUFF 64

and add bellow the next line:

#define _SS_MAX_RX_BUFF 128

You just increased the RX buffer size for UNO and other "snails".

Now you can compile and upload your code supporting highier buffered serial input data.
*/

//next 2 definition: leave them commented for standard conectivity over Software serial
//#define usejLader                     //un-comment this if you use micro and nano GSM 3G adapter for ArduinoNano --Do not use it with b-gsmgnss!!!!
//#define HARDWARESERIAL                //remove comment to use Serial1 for communication on AT MEGA 2560...DUE..
```

```
49
50  //#define atDebug                       //un-comment this if you want to debug the
    AT exchange
51
52  /*next one - change general read buffer(buffd) size 1024 uncommented, 256 commented */
53  char message[]="Hello world!";                //test.tst content - no more than
    1024(128 in UNO_MODE) chars! see buffd size, bellow
54
55
56  /*do not change under this line! Insteed, make one copy for playing with.*/
57  #define powerPIN        7//Arduino Digital pin used to power up / power down the modem
58  #define resetPIN        6//Arduino Digital pin used to reset the modem
59  #define statusPIN       5//Arduino Digital pin used to monitor if modem is powered
60
61  #if (ARDUINO >= 100)
62    #include "Arduino.h"
63      #if !defined(HARDWARESERIAL)
64          #include <SoftwareSerial.h>
65      #endif
66  #else
67    #include "WProgram.h"
68      #if !defined(HARDWARESERIAL)
69          #include <NewSoftSerial.h>
70      #endif
71  #endif
72
73  #if defined(HARDWARESERIAL)
74      #define BUFFDSIZE 1024
75  #else
76      #if defined(__AVR_ATmega1280__) /*AT MEGA ADK*/|| defined(__AVR_ATmega2560__)
        /*AT MEGA 2560*/|| defined(__AVR_ATmega32U4__) /*LEONARDO*/
77          SoftwareSerial agsmSerial(10,3);  //RX==>10,TX soft==>3...read
78          #define BUFFDSIZE 1024
79      #else/*UNO*/
80          #define UNO_MODE  //Arduino UNO
81          #define BUFFDSIZE 200 //240
82          #if defined usejLader
83              SoftwareSerial agsmSerial(3, 2);  //RX==>3 ,TX soft==>2
84          #else
85              SoftwareSerial agsmSerial(2, 3);  //RX==>2 ,TX soft==>3
86          #endif
87      #endif
88  #endif
89
90
91  #include "bgsmgnss_FS_lbr.h"
92
93  #define printDebugLN(x){Serial.println(x);}
94
95  #if defined(UNO_MODE)
96    char buffd[256];
97  #else
98    char buffd[512];
99  #endif
100
101 int state=0, i=0, powerState = 0;
102 char ch;
103 //char IMEI[18];
104 unsigned long offsetTime;
105 int totalChars = 0;
106 int ready4SMS = 0;
107 int ready4Voice = 0;
```

```
108  //char buffd[BUFFDSIZE];
109  //int noSMS=0, totSMS=0;
110  char readBuffer[200];
111
112
113  void setup(){
114      agsmSerial.begin(9600);
115      Serial.begin(57600);
116      clearagsmSerial();
117      clearSerial();
118      delay(10);
119
120      modemHWSetup();                          //configure Arduino IN and OUT to be
         used with modem
121
122      Serial.flush();
123      agsmSerial.flush();
124      delay(1000);
125      Serial.println(F("b-gsmgnss SD read/write/delete file example"));
126      Serial.flush();
127
128      Serial.println(F("seat back and relax until b-gsmgnss is ready"));
129      delay(100);
130
131      powerOnModem();
132
133      clearBUFFD();
134      while(strlen(buffd)<1){
135          getIMEI();
136          delay(500);
137      }
138
139      ready4SMS = 0;
140      ready4Voice = 0;
141
142      Serial.println(F("b-gsmgnss ready.. let's run the example"));
143      Serial.print(F("b-gsmgnss IMEI: ")); Serial.flush();
144      Serial.println(buffd); Serial.flush();
145      //setAUDIOchannel(20);
146      delay(1000);
147  }
148
149
150  void loop(){
151    int res;
152
153    switch(state){
154      case 0://check modem status
155        if(!getModemState()) restartMODEM();
156        else
157          state++;
158        i=0;
159        res= 0;
160        while(res != 1){
161          res = sendATcommand("","OK","ERROR",2);
162          if (res != 1) {
163            if(i++ >= 10) {
164              printDebugLN(F("AT err...restarting"));
165              restartMODEM();
166            }
167          }
168          delay(500);
```

```
169         }
170         sendATcommand("+IPR=0;&w","OK","ERROR",2);
171         delay(2000);
172       break;
173 
174       case 1:
175         clearBUFFD();
176         //next some init strings...
177         aGsmCMD("AT+QIMODE=0",200);
178         aGsmCMD("AT+QINDI=0",200);
179         aGsmCMD("AT+QIMUX=0",200);
180         aGsmCMD("AT+QIDNSIP=0",200);
181         offsetTime=0;
182         clearBUFFD();
183         state++;
184         break;
185 
186       case 2:
187         printDebugLN(F("try CPIN..."));
188         if(!offsetTime) offsetTime = millis();
189         if ((millis() - offsetTime) > 20000) restartMODEM();
190         if(sendATcommand("+CPIN?","READY")==1){
191           offsetTime=0; state++;
192           printDebugLN(F("READY"));
193         }else{}
194         clearagsmSerial(); delay(100);
195         offsetTime = millis();
196         break;
197 
198       case 3:
199         if(!offsetTime) offsetTime = millis();
200         if ((millis() - offsetTime) > 30000) restartMODEM();
201 
202         printDebugLN(F("Query GSM registration?"));
203         res = registration(GSM);
204         if(res==1){
205           offsetTime=0; state++;
206           printDebugLN(F("READY, HOME NETWORK"));
207         }else if(res==5){
208           offsetTime=0; state++;
209           printDebugLN(F("READY, ROAMING"));
210         }else{
211           Serial.print(F("."));
212         }
213         offsetTime = millis();
214       break;
215 
216       case 4: //init SIM/MODEM
217         printDebugLN(F("Query State of Initialization"));
218         if(sendATcommand("+QINISTAT","3")==1){
219           offsetTime=0; state++;
220           printDebugLN(F("READY"));
221         }else{Serial.print(F(".")); delay(100);}
222         clearagsmSerial(); delay(100);
223         offsetTime = millis();
224       break;
225 
226       case 5://list files on SD
227       //case 7:
228         if(!offsetTime) offsetTime = millis();
229         if ((millis() - offsetTime) > 5000) restartMODEM();
230 
```

```
231         printDebugLN(F("list all text files")); delay(100);
232         listModemFile("*.txt");
233         printDebugLN(buffd);
234         delay(1000);
235 
236         printDebugLN(F("The file system it is NOT case SENSITIVE!")); delay(100);
237         delay(10000);
238 
239         state++;
240       break;
241 
242       case 6:
243         printDebugLN(F("Try to read inexistent test.TXT!")); delay(100);
244         clearBUFFD();
245         memset(readBuffer,0x00, sizeof(readBuffer));
246         if(readModemFile("test.TXT", readBuffer)>0) {
247               printDebugLN(readBuffer);
248         }else printDebugLN(F("Error reading file"));
249         memset(readBuffer,0x00, sizeof(readBuffer));
250         clearBUFFD();
251 
252         delay(10000);
253 
254         printDebugLN(F("Try to write test.txt with defined message")); delay(100);
255         if(writeModemFile("test.txt", message)>0) {
256           printDebugLN(F("Success writing file!"));
257         }else printDebugLN(F("Error writing file!"));
258 
259         delay(10000);
260 
261         clearBUFFD();
262         printDebugLN("now list text files"); delay(100);
263         listModemFile("*.txt");
264         printDebugLN(buffd);
265 
266         delay(10000);
267 
268         clearBUFFD();
269         printDebugLN(F("Now read test.txt!")); delay(100);
270         memset(readBuffer,0x00, sizeof(readBuffer));
271         readModemFile("test.TXT", readBuffer);
272         printDebugLN(readBuffer); delay(10000);
273 
274         clearBUFFD();
275         printDebugLN(("Now delete test.txt!")); delay(100);
276         deleteModemFile("test.txt");
277         delay(5000);
278 
279         printDebugLN(F("now list text files")); delay(100);
280         listModemFile("*.txt");
281         printDebugLN(buffd); delay(10000);
282 
283         delay(2000);
284 
285         printDebugLN(F("That's all folks!"));
286         state++;
287       break;
288 
289       case 7:
290       default:
291         //restartMODEM();
292         delay(1000000);
```

```
293         state=0;
294       break;
295     }
296 
297   }
298 
```