```
/*****************************************************************************
* Arduino library for the Devantech LCD03/LCD05 with 3x4 keypad - V2.4 (ANSI C) *
* https://www.robot-electronics.co.uk/htm/Lcd03tech.htm                       *
* © Copyright 2019-2020 Filippo Pardini - filippo@robotica.eng.br             *
* http://blog.robotica.eng.br/                                                *
*                                                                             *
* This program is free software: you can redistribute it and/or modify it     *
* under the terms of the GNU Lesser General Public License as published by    *
* the Free Software Foundation, either version 3 of the License, or any       *
* later version.                                                              *
*                                                                             *
* This program is distributed 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. See the                *
* GNU Lesser General Public License for more details.                         *
*                                                                             *
* You should have received a copy of the GNU Lesser General Public License    *
* along with this program. If not, see <http://www.gnu.org/licenses/>.        *
*****************************************************************************
*                                                                             *
* Remember that the TWI bus must have pull-up resistors on SDA and SCL.       *
* As LCD03/LCD05 are 5V, if the microcontroller is 3.3V a logic level         *
* converter must be used. I recommend 1.8K or 4.7K resistors.                 *
*                                                                             *
* This version was tested with Arduino 1.8.9 IDE and ESP-WROOM-32             *
* DOIT ESP32 DEVKIT V1 board. Some boards, as that, already has pull-up       *
* resistors included at the I2C default pins (in this case SDA 21 SCL 22)     *
*                                                                             *
*****************************************************************************/

#ifndef LCD03_N1_h
#define LCD03_N1_h

//_____________________________________________________________________________________________________________

char endereco;                                          // Endereço LCD03
unsigned long tespera;                                  // Tempo de espera máximo no teclado (seg)

//_____________________________________________________________________________________________________________

#define cursor_home envia_comando(1);
#define hide_cursor envia_comando(4);
#define show_underline_cursor envia_comando(5);
#define show_blinking_cursor envia_comando(6);
#define backspace envia_comando(8);
#define horizontal_tab envia_comando(9);
```

```
47  #define smart_line_feed envia_comando(10);
48  #define vertical_tab envia_comando(11);
49  #define clear_screen envia_comando(12);
50  #define clear_line(li) display_texto_lc("                    ",li,1);
51  #define carriage_return envia_comando(13);
52  #define clear_column envia_comando(17);
53  #define tab_set envia_comando(18);
54  #define backlight_on envia_comando(19);
55  #define backlight_off envia_comando(20);
56  #define disable_startup_message envia_comando(21);
57  #define enable_startup_message envia_comando(22);
58  #define double_keypad_scan_rate envia_comando(28);
59  #define normal_keypad_scan_rate envia_comando(29);
60  #define change_address(adr) muda_endereco(adr);                    // adr => 0xC6 ou 0xC8 ou 0xCA ou 0xCC ou 0xCE
61
62  //_____________________________________________________________________________________________________________
63
64  void set_cursor(byte pos);                                         // Posiciona cursor 1 a 80
65  void set_cursor_lc(byte linha,byte coluna);                        // Posiciona cursor na linha e coluna
66  void envia_comando(byte cmd);                                      // Envia um comando
67  void muda_endereco(byte ender);                                    // Muda o endereço do LCD03
68  byte read_keypad(void);                                            // Le o caractere do teclado
69  void lcd_registers_read(byte *fifo,byte *lsb,byte *msb,byte *ver); // Le os 4 registros do LCD03
70  char convert_keypad(byte tipo,byte car);                           // Converte os bits do teclado para caracteres
71  bool entrada_teclado(char *str);                                   // Processa a entrada no teclado
72  void LCD03_begin(char ender,unsigned long tes);                    // Inicializa o LCD03
73  void Serial_print(char *vetor,size_t inicio,size_t cp);            // Imprime um vetor char no monitor serial
74  void display_float_lc(float flt,uint8_t lin,uint8_t col);          // Display de um float na linha e coluna
75  void display_float(float flt);                                     // Display de um float onde está o cursor
76  void display_texto_lc(const char *txt,uint8_t lin,uint8_t col);    // Display de um texto na linha e coluna
77  void display_texto(const char *txt);                               // Display de um texto onde está o cursor
78  void display_vectorchar_lc(char *data,uint8_t lin,uint8_t col);    // Display de um vetor char na linha e coluna
79  void display_vectorchar(char *data);                               // Display de um vetor char onde está o cursor
80  void display_char_lc(char val,uint8_t lin,uint8_t col);            // Display de um caractere na linha e coluna
81  void display_char(char val);                                       // Display de um caractere onde está o cursor
82  void display_string_lc(String *stri,uint8_t lin,uint8_t col);      // Display de um String na linha e coluna
83  void display_string(String *stri);                                 // Display de um String onde está o cursor
84
85  //_____________________________________________________________________________________________________________
86
87  void LCD03_begin(char ender,unsigned long tes)
88  {
89    // Inicializa o LCD03
90
91    byte buf;
92    byte lsb;
```

```
93    byte msb;
94    byte ver;
95
96    endereco = ender >> 1;
97    tespera = tes * 1000;
98    lcd_registers_read(&buf,&lsb,&msb,&ver);                    // Lê os 4 registros
99    Serial.print("\nLCD03 versao ");
100   Serial.print(ver,DEC);
101   Serial.print("\nFIFO buffer = ");
102   Serial.print(buf,DEC);
103   Serial.print(" Bytes\n");
104 }
105
106 //_____________________________________________________________________________________________________
107
108 void set_cursor(byte pos)
109 {
110   // Posiciona o cursor 1 <= pos <= 80
111
112   byte dad[3];
113
114   dad[0] = 0;
115   dad[1] = 2;
116   dad[2] = pos;
117   Wire.beginTransmission(endereco);
118   Wire.write(dad,3);
119   Wire.endTransmission();
120 }
121
122 //_____________________________________________________________________________________________________
123
124 void set_cursor_lc(byte linha,byte coluna)
125 {
126   // Posiciona o cursor na linha e coluna
127
128   byte dad[4];
129
130   dad[0] = 0;
131   dad[1] = 3;
132   dad[2] = linha;
133   dad[3] = coluna;
134   Wire.beginTransmission(endereco);
135   Wire.write(dad,4);
136   Wire.endTransmission();
137 }
138
```

```
139 //_____________________________________________________________________________________________________________
140
141 void envia_comando(byte cmd)
142 {
143   // Envia um comando para o registro 0
144
145   byte dad[2];
146
147   dad[0] = 0;
148   dad[1] = cmd;
149   Wire.beginTransmission(endereco);
150   Wire.write(dad,2);
151   Wire.endTransmission();
152 }
153
154 //_____________________________________________________________________________________________________________
155
156 void muda_endereco(byte ender)
157 {
158   // Muda endereço I2C
159
160   byte dad[7];
161
162   dad[0] = 0;
163   dad[1] = 25;
164   dad[2] = 0x19;
165   dad[3] = 0xA0;
166   dad[4] = 0xAA;
167   dad[5] = 0xA5;
168   dad[6] = ender;                                          // 0xC6 ou 0xC8 ou 0xCA ou 0xCC ou 0xCE
169   Wire.beginTransmission(endereco);
170   Wire.write(dad,7);
171   Wire.endTransmission();
172 }
173
174 //_____________________________________________________________________________________________________________
175
176 void lcd_registers_read(byte *fifo,byte *lsb,byte *msb,byte *ver)
177 {
178   // Lê os 4 registros do LCD03
179
180   Wire.requestFrom(endereco,4);
181   *fifo = Wire.read();                                     // Registro 0 - fifo buffer
182   *lsb = Wire.read();                                      // Registro 1 - lsb teclado
183   *msb = Wire.read();                                      // Registro 2 - msb teclado
184   *ver = Wire.read();                                      // Registro 3 - Versão
```

```
185  }
186
187  //_____________________________________________________________________________________________________
188
189  bool entrada_teclado(char *str)
190  {
191    // Lê o teclado e recebe um vetor de caracteres. Nao é testado o espaco livre no buffer FIFO de 64 bytes
192    // A leitura é encerrada quando é digitado #
193
194    byte cod;                                                    // Variável para ler caractere do teclado
195    uint8_t i;                                                   // Indice
196    unsigned long t;                                             // Contador de tempo
197
198    str[0] = ' ';                                                // Primeiro elemento sempre ' '
199    i = 1;
200    cod = 0;                                                     // Valor inicial inválido
201
202    // Espera no máximo "tespera" milisegundos para teclar
203    t = millis();                                                // Inicia contagem de tempo
204    while((cod == 0) && ((millis() - t) < tespera)) cod = read_keypad();
205    if (cod == 0) return false;                                  // Esgotou o tempo
206    else                                                         // Tem caractere
207    {
208      str[i++] = cod;                                            // Armazena no string
209      while(1)                                                   // Até entrar com #
210      {
211        while(read_keypad() != 0);                               // Espera reset dos registros para 0
212        while((cod = read_keypad()) == 0);                       // Espera proxima digitação
213        if(cod == '#')                                           // Se for #
214        {
215          str[i] = '\0';                                         // Encerra o string
216          return true;                                           // Volta OK
217        }
218        else                                                     // Não é #, é outro caractere
219        {
220          str[i++] = cod;                                        // Armazena no string
221        }
222      }
223    }
224  }
225
226  //_____________________________________________________________________________________________________
227
228  byte read_keypad(void)
229  {
230    // Le o caractere digitado no teclado
```

```
231
232     byte fifo,lsb,msb,ver;
233     byte caractere;
234
235     lcd_registers_read(&fifo,&lsb,&msb,&ver);
236     if (lsb != 0) caractere = convert_keypad(1,lsb);
237     else if (msb != 0) caractere = convert_keypad(2,msb);
238     else caractere = 0;
239     return caractere;
240   }
241
242   //_____________________________________________________________________________________________________
243
244   char convert_keypad(byte tipo,byte car)
245   {
246     // Converte os bits do teclado para caracteres
247
248     byte baite;
249
250     baite = 0;      // Inválido
251     switch (tipo)
252     {
253       case 1:       // É LSB
254       {
255         switch (car)
256         {
257           case 1:     {baite = 49; break;}    // ASCII 1
258           case 2:     {baite = 50; break;}    // ASCII 2
259           case 4:     {baite = 51; break;}    // ASCII 3
260           case 8:     {baite = 52; break;}    // ASCII 4
261           case 16:    {baite = 53; break;}    // ASCII 5
262           case 32:    {baite = 54; break;}    // ASCII 6
263           case 64:    {baite = 55; break;}    // ASCII 7
264           case 128:   {baite = 56; break;}    // ASCII 8
265           default:    break;
266         }
267         break;
268       }
269
270       case 2  :   // É MSB
271       {
272         switch (car)
273         {
274           case 1:     {baite = 57; break;}    // ASCII 9
275           case 2:     {baite = 42; break;}    // ASCII *
276           case 4:     {baite = 48; break;}    // ASCII 0
```

```
277           case 8:     {baite = 35; break;}    // ASCII #
278           default:    break;
279         }
280         break;
281       }
282       default:    break;
283     }
284     return baite;
285   }
286
287   //_____________________________________________________________________________________________________
288
289   void Serial_print(char *vetor,size_t inicio,size_t cp)
290   {
291     // Imprime no monitor serial um vetor de bytes iniciando na posição inicio por cp posições
292
293     size_t i,k;
294     char crt;
295
296     k = inicio + cp;
297     for(i=inicio;i<k;i++)
298     {
299       switch (vetor[i])
300       {
301         case 49:  {crt = '1'; break;}    // ASCII 1
302         case 50:  {crt = '2'; break;}    // ASCII 2
303         case 51:  {crt = '3'; break;}    // ASCII 3
304         case 52:  {crt = '4'; break;}    // ASCII 4
305         case 53:  {crt = '5'; break;}    // ASCII 5
306         case 54:  {crt = '6'; break;}    // ASCII 6
307         case 55:  {crt = '7'; break;}    // ASCII 7
308         case 56:  {crt = '8'; break;}    // ASCII 8
309         case 57:  {crt = '9'; break;}    // ASCII 9
310         case 42:  {crt = '*'; break;}    // ASCII *
311         case 48:  {crt = '0'; break;}    // ASCII 0
312         case 35:  {crt = '#'; break;}    // ASCII #
313       }
314       Serial.print(crt);
315     }
316   }
317
318   //_____________________________________________________________________________________________________
319
320   void display_float_lc(float flt,uint8_t lin,uint8_t col)
321   {
322     // Faz o display de um float com 2 decimais a partir da linha lin coluna col
```

```
323
324     String stri;
325
326     stri = String(flt,2);
327     set_cursor_lc(lin,col);
328     Wire.beginTransmission(endereco);
329     Wire.write(stri.c_str());
330     Wire.endTransmission();
331   }
332
333   //_____________________________________________________________________________________________________
334
335   void display_float(float flt)
336   {
337     // Faz o display de um float com 2 decimais a partir da posição do cursor
338
339     String stri;
340
341     Wire.beginTransmission(endereco);
342     Wire.write(stri.c_str());
343     Wire.endTransmission();
344   }
345
346   //_____________________________________________________________________________________________________
347
348   void display_texto_lc(const char *txt,uint8_t lin,uint8_t col)
349   {
350     // Faz o display de um texto a partir da linha lin coluna col
351
352     set_cursor_lc(lin,col);
353     Wire.beginTransmission(endereco);
354     Wire.write(txt);
355     Wire.endTransmission();
356   }
357
358   //_____________________________________________________________________________________________________
359
360   void display_texto(const char *txt)
361   {
362     // Faz o display de um texto a partir da posição do cursor
363
364     Wire.beginTransmission(endereco);
365     Wire.write(txt);
366     Wire.endTransmission();
367   }
368
```

```
369 //_____________________________________________________________________________________________________
370
371 void display_vectorchar_lc(char *data,uint8_t lin,uint8_t col)
372 {
373   // Faz o display de um vetor de caracteres a partir da linha lin coluna col
374
375   String stri;
376
377   stri = String(*data);
378   set_cursor_lc(lin,col);
379   Wire.beginTransmission(endereco);
380   Wire.write(stri.c_str());
381   Wire.endTransmission();
382 }
383
384 //_____________________________________________________________________________________________________
385
386 void display_vectorchar(char *data)
387 {
388   // Faz o display de um vetor de caracteres a partir da posição do cursor
389
390   String stri;
391
392   stri = String(*data);
393   Wire.beginTransmission(endereco);
394   Wire.write(stri.c_str());
395   Wire.endTransmission();
396 }
397
398 //_____________________________________________________________________________________________________
399
400 void display_char_lc(char val,uint8_t lin,uint8_t col)
401 {
402   // Faz o display de um caractere a partir da linha lin coluna col
403
404   set_cursor_lc(lin,col);
405   Wire.beginTransmission(endereco);
406   Wire.write(val);
407   Wire.endTransmission();
408 }
409
410 //_____________________________________________________________________________________________________
411
412 void display_char(char val)
413 {
414   // Faz o display de um caractere a partir da posição do cursor
```

```
415
416     Wire.beginTransmission(endereco);
417     Wire.write(val);
418     Wire.endTransmission();
419   }
420
421   //_____________________________________________________________________________________________
422
423   void display_string_lc(String *stri,uint8_t lin,uint8_t col)
424   {
425     // Faz o display de um String a partir da linha lin coluna col
426
427     set_cursor_lc(lin,col);
428     Wire.beginTransmission(endereco);
429     Wire.write(stri->c_str());
430     Wire.endTransmission();
431   }
432
433   //_____________________________________________________________________________________________
434
435   void display_string(String *stri)
436   {
437     // Faz o display de um String a partir da posição do cursor
438
439     Wire.beginTransmission(endereco);
440     Wire.write(stri->c_str());
441     Wire.endTransmission();
442   }
443
444   //_____________________________________________________________________________________________
445
446
447   #endif
448
```