```c
/* ************************************************************************** */
/*                                                                            */
/*   Nome do arquivo:       ledSwi.c                                          */
/*                                                                            */
/*   Descrição:             Arquivo contendo as funcoes que lidam com         */
/*                       a atuacao do microcontrolador com os LEDs e          */
/*                       botoes do kit                                        */
/*                                                                            */
/*   Autores:              Gustavo Lino e Giácomo Dollevedo                   */
/*   Criado em:             31/03/2020                                        */
/*   Ultima revisão em:     31/07/2020                                        */
/* ************************************************************************** */


/*Adequações realizadas: Não deve ser considerada as variaveis tipo booleanas, para isso foram trocadas por tipo char
outra adequação implementada foi o testar se os parametros passados estão dentro do padrão esperado*/


#include "ledSwi.h"
#include "board.h"


/* ************************************************************************** */
/* Nome do metodo:         ledSwi_init                                        */
/* Descrição:              Inicializa os clocks e pinos necessarios para utilizar */
/*      a interface de botoes/leds do kit                                     */
/*                                                                            */
/* Parametros de entrada:   5 char (0 ou 1) que indica se o pino sera configurado   */
/*      como led ou como botao                                                */
/*                      0 -> botao;  1 -> led                                 */
/*                                                                            */
/* Parametros de saida:     n/a                                               */
/* ************************************************************************** */
char cLedSwi1 = 0, cLedSwi2 = 0, cLedSwi3 = 0, cLedSwi4 = 0;
unsigned char ucError = 0;

void ledSwi_init(char led1, char led2, char led3, char led4) {



/* ativar o clock para a porta A*/
   SIM_SCGC5 |= uiSetClockPort;

/*Configura os pinos das portas para GPIO*/

   PORTA_PCR1 |= uiSetPinAsGPIO;
   PORTA_PCR2 |= uiSetPinAsGPIO;
   PORTA_PCR4 |= uiSetPinAsGPIO;
   PORTA_PCR5 |= uiSetPinAsGPIO;

 // testa se os parametros foram passado corretamente, se nao foram ativa todas as portas como led

 if((led1 != 0 && led1 != 1) && (led2 != 0 && led2 != 1) && (led3 != 0 && led3 != 1) && (led4 != 0 && led4 != 1)) {
 GPIOA_PDDR |= uiPin1MaskEnable;
      cLedSwi1 = 1;
      GPIOA_PDDR |= uiPin2MaskEnable;
      cLedSwi2 = 1;
      GPIOA_PDDR |= uiPin3MaskEnable;
      cLedSwi3 = 1;
      GPIOA_PDDR |= uiPin4MaskEnable;
      cLedSwi4 = 1;
 }

/*Define se os pinos serao entrada (chave) ou saida (led)*/

   if(led1 == 1){
```

```
64      GPIOA_PDDR |= uiPin1MaskEnable;
65      cLedSwi1 = 1;
66    }
67    else{
68      GPIOA_PDDR &= uiPin1MaskDisable;
69      cLedSwi1 = 0;
70    }
71
72    if(led2 == 1){
73      GPIOA_PDDR |= uiPin2MaskEnable;
74      cLedSwi2 = 1;
75    }
76    else{
77      GPIOA_PDDR &= uiPin2MaskDisable;
78      cLedSwi2 = 0;
79    }
80
81    if(led3== 1){
82      GPIOA_PDDR |= uiPin4MaskEnable;
83      cLedSwi3 = 1;
84    }
85    else{
86      GPIOA_PDDR &= uiPin4MaskDisable;
87      cLedSwi3 = 0;
88    }
89
90    if(led4 == 1){
91      GPIOA_PDDR |= uiPin5MaskEnable;
92      cLedSwi4 = 1;
93    }
94    else{
95      GPIOA_PDDR &= uiPin5MaskDisable;
96      cLedSwi4 = 0;
97    }
98
99  }
100
101
102 /* ************************************************************************** */
103 /* Nome do metodo:          readSwitch                                        */
104 /* Descrição:               Le o status de um switch para saber se o mesmo    */
105 /*                          está pressionado ou não                           */
106 /*                                                                            */
107 /* Parametros de entrada:   Um inteiro (0<n<5) que indica qual botão será lido */
108 /*                          inicializado como entrada (botao) ou saida (LED)  */
109 /*                          0 -> Leitura PTA1; 1 -> Leitura PTA2,             */
110 /*                          2 -> Leitura PTA4; 3 -> Leitura PTA5;             */
111 /*                                                                            */
112 /* Parametros de saida:     Um char indicando se o botao lido está sendo      */
113 /*                          pressionado (1), se não está ou se é inválido     */
114 /*                          (0)                                               */
115 /* ************************************************************************** */
116 char readSwitch(int n){
117 //testa se houve erro na passagem dos parametros
118  if(n > 5){
119   ucError = 1;
120  }
121
122  else{
123   ucError = 0;
124  }
125
126 // se não houver erro na entrada
127  if(0 == ucError){
128
129
130
```

```
131     switch(n){
132       case 1:
133         if(0 == cLedSwi1){
134           if (uiPin1MaskEnable == (GPIOA_PDIR & uiPin1MaskEnable)){
135             return 1;
136           }
137           else {
138             return 0;
139           }
140         }
141         else{
142           return 0;
143         }
144       break;
145 
146       case 2:
147         if(0 == cLedSwi2){
148           if (uiPin2MaskEnable == (GPIOA_PDIR & uiPin2MaskEnable)){
149             return 1;
150           }
151           else {
152             return 0;
153           }
154         }
155         else{
156           return 0;
157         }
158       break;
159 
160       case 3:
161         if(0 == cLedSwi3){
162           if (uiPin4MaskEnable == (GPIOA_PDIR & uiPin4MaskEnable)){
163             return 1;
164           }
165           else {
166             return 0;
167           }
168         }
169         else{
170           return 0;
171         }
172       break;
173 
174       case 4:
175         if(0 == cLedSwi4){
176           if (uiPin5MaskEnable == (GPIOA_PDIR & uiPin5MaskEnable)){
177             return 1;
178           }
179           else {
180             return 0;
181           }
182         }
183       else{
184             return 0;
185         }
186       break;
187       }
188     }
189   return 0;
190    }
191 
192 
193 
194 /* ************************************************************************** */
195 /* Nome do metodo:          writeLED                                          */
196 
197 
```

```
198 /* Descrição:              Liga ou desliga o LED selecionado conforme as          */
199 /*                        entradas                                               */
200 /*                                                                               */
201 /* Parametros de entrada:   Um inteiro (0<n<5) indicando sobre qual LED sera      */
202 /*                        efetuado o comando;                                    */
203 /*                        Um char (status) indicando se o LED sera               */
204 /*                        aceso (1) ou apagado (0)                               */
205 /*                                                                               */
206 /* Parametros de saida:     n/a                                                  */
207 /* ****************************************************************************** */
208 void writeLED(int n, char status){
209
210 //testa se houve erro na passagem dos parametros
211
212  if((status != 0 && status != 1) && (n < 5) ){
213   ucError = 1;
214  }
215
216  else{
217   ucError = 0;
218  }
219
220 // se não houver erro na entrada
221  if (0 == ucError){
222
223     switch(n){
224
225         case 1:
226             if(1 == cLedSwi1){
227                 if (status){
228                     GPIOA_PDOR |= uiPin1MaskEnable;
229                 }
230                 else {
231                     GPIOA_PDOR |= uiPin1MaskDisable;
232                 }
233             }
234
235         break;
236
237         case 2:
238             if(1 == cLedSwi2){
239                 if (status){
240                     GPIOA_PDOR |= uiPin2MaskEnable;
241                 }
242                 else {
243                     GPIOA_PDOR |= uiPin2MaskDisable;
244                 }
245             }
246
247         break;
248
249         case 3:
250             if(1 == cLedSwi3){
251                 if (status){
252                     GPIOA_PDOR |= uiPin4MaskEnable;
253                 }
254                 else {
255                     GPIOA_PDOR |= uiPin4MaskDisable;
256                 }
257             }
258
259         break;
260
261         case 4:
262             if(1 == cLedSwi4){
263                 if (status){
264
```

```
                GPIOA_PDOR |= uiPin5MaskEnable;
            }
            else {
                GPIOA_PDOR |= uiPin5MaskDisable;
            }
          }

        break;
     }
    }
}


/* ************************************************************************** */
/* Nome do metodo:          turnOnLED                                         */
/* Descrição:               Liga um LED especificado pela entrada             */
/*                                                                            */
/* Parametros de entrada:   Um inteiro (0<n<5) indicando qual LED sera aceso  */
/*                                                                            */
/* Parametros de saida:     n/a                                               */
/* ************************************************************************** */
void turnOnLED(int n){
//testa se houve erro na passagem dos parametros
 if(n > 5){
  ucError = 1;
 }

 else{
  ucError = 0;
 }

// se não houver erro na entrada
 if(0 == ucError){
    switch(n){

        case 1:
            if(1 == cLedSwi1){
                GPIOA_PSOR |= uiPin1MaskEnable;
            }
        break;

        case 2:
            if(1 == cLedSwi2){
                GPIOA_PSOR |= uiPin2MaskEnable;
            }
        break;

        case 3:
            if(1 == cLedSwi3){
                GPIOA_PSOR |= uiPin4MaskEnable;
            }
        break;

        case 4:
            if(1 == cLedSwi4){
                GPIOA_PSOR |= uiPin5MaskEnable;
            }
        break;
     }

   }
}


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

```
/* Nome do metodo:          turnOffLED                                      */
/* Descrição:               Desliga um LED especificado pela entrada        */
/*                                                                          */
/* Parametros de entrada:   Um inteiro (0<n<5) indicando qual LED sera apagado  */
/*                                                                          */
/* Parametros de saida:     n/a                                             */
/* ************************************************************************ */
void turnOffLED(int n){
//testa se houve erro na passagem dos parametros
 if(n > 5){
  ucError = 1;
 }

 else{
  ucError = 0;
 }

// se não houver erro na entrada
 if(0 == ucError){

    switch(n){

        case 1:
            if(1 == cLedSwi1){
                GPIOA_PCOR |= uiPin1MaskDisable;
            }
        break;

        case 2:
            if(1 == cLedSwi2){
                GPIOA_PCOR |= uiPin2MaskDisable;
            }
        break;

        case 3:
            if(1 == cLedSwi3){
                GPIOA_PCOR |= uiPin4MaskDisable;
            }
        break;

        case 4:
            if(1 == cLedSwi4){
                GPIOA_PCOR |= uiPin5MaskDisable;
            }
        break;
     }

   }
}



/* ************************************************************************ */
/* Nome do metodo:          toggleLED                                       */
/* Descrição:               Inverte o status atual de um LED especificado pela  */
/*                          entrada                                         */
/*                                                                          */
/* Parametros de entrada:   Um inteiro (0<n<5) indicando qual LED tera seu      */
/*                          status invertido                                */
/*                                                                          */
/* Parametros de saida:     n/a                                             */
/* ************************************************************************ */
void toggleLED(int n){
//testa se houve erro na passagem dos parametros
 if(n > 5){
  ucError = 1;
```

```
399  }
400
401  else{
402   ucError = 0;
403  }
404
405 // se não houver erro na entrada
406  if(0 == ucError){
407
408     switch(n){
409
410         case 1:
411             if(1 == cLedSwi1){
412                 GPIOA_PTOR |= uiPin1MaskEnable;
413             }
414         break;
415
416         case 2:
417             if(1 == cLedSwi2){
418                 GPIOA_PTOR |= uiPin2MaskEnable;
419             }
420         break;
421
422         case 3:
423             if(1 == cLedSwi3){
424                 GPIOA_PTOR |= uiPin4MaskEnable;
425             }
426         break;
427
428         case 4:
429             if(1 == cLedSwi4){
430                 GPIOA_PTOR |= uiPin5MaskEnable;
431             }
432         break;
433     }

    }
}
```