```c
/*****************************************************************************
*
 * HEIG-VD
 * Haute Ecole d'Ingenerie et de Gestion du Canton de Vaud
 * School of Business and Engineering in Canton de Vaud

 *****************************************************************************
 *
 * REDS Institute
 * Reconfigurable Embedded Digital Systems

 *****************************************************************************
 *
 *
 * File                 : labo5.c
 * Author               : Spinelli Isaia
 * Date                 : 01.05.2020
 *
 * Context              : SOCF tutorial lab
 *

 *****************************************************************************
 *
 * Brief: Programme for labo 5 of SOCF, for DE1-SoC board
 *
 *

 *****************************************************************************
 *
 * Modifications :
 * Ver    Date        Student      Comments
 * 0.1    01.05.20    Isaia Spinelli : Modif pour la partie 1
 * 1.1    03.05.20    Isaia Spinelli : Ajout de la partie 2
 * 1.2    08.05.20    Isaia Spinelli : Test du paramètre edge des irq
*****************************************************************************
/

#include "defines.h"


/* Variable globales */

int irqKey2 = 0;
int irqKey3 = 0;




int main(void){

    // tableau de converssion         0     1     2     3    4      5    6     7     8     9    a      b    c      d     e    f
    char tab_dec_to_hex_7seg[16] = {0x40, 0xF9, 0x24, 0x30, 0x19, 0x12, 0x02, 0xF8, 0x00, 0x10, 0x08, 0x03, 0x27, 0x21, 0x06, 0x0e };
    int led_tmp,Seg_tmp;


    /*---------- INTI ----------*/
    AXI_HEX5 = 0x40;
    AXI_HEX4 = 0xF9;
    AXI_HEX3 = 0x24;
    AXI_HEX2 = 0x30;
    AXI_HEX1 = 0x19;
    AXI_HEX0 = 0x02;

    AXI_LEDS = AXI_SWITCHES;

    unsigned int cst = AXI_REG_CONST;
    AXI_REG_TEST = cst;
```

```
58
59      // Masque le bouton key3 (pour tester le masquage des interruptions)
60      // AXI_INT_MASK = KEY3;
61
62      // KEY 3 sur flanc montant
63      AXI_INT_EDGE = KEY3;
64
65      disable_A9_interrupts();   // disable interrupts in the A9 processor
66      set_A9_IRQ_stack();        // initialize the stack pointer for IRQ mode
67      config_GIC();              // configure the general interrupt controller
68      config_KEYs();             // configure KEYs to generate interrupts
69      enable_A9_interrupts();    // enable interrupts in the A9 processor
70
71
72
73
74      while(1){
75          /* Appuie sur KEY 0*/
76          if ((AXI_KEYS & KEY0) == 0) {
77              // l'états des switches est copiés sur les LEDs.
78              AXI_LEDS = AXI_SWITCHES;
79              // Les afficheurs HEX5 à HEX0 affichent en hexadécimal les bits 23 à 0 de
                la constante définie dans l'IP.
80              AXI_HEX0 = tab_dec_to_hex_7seg[cst & 0xF];
81              AXI_HEX1 = tab_dec_to_hex_7seg[(cst>>4)  & 0xF];
82              AXI_HEX2 = tab_dec_to_hex_7seg[(cst>>8)  & 0xF];
83              AXI_HEX3 = tab_dec_to_hex_7seg[(cst>>12) & 0xF];
84              AXI_HEX4 = tab_dec_to_hex_7seg[(cst>>16) & 0xF];
85              AXI_HEX5 = tab_dec_to_hex_7seg[(cst>>20) & 0xF];
86
87
88          /* Appuie sur KEY 1 */
89          } else if ((AXI_KEYS & KEY1) == 0) {
90              //  l'états inverses des switches est copiés sur les LEDs.
91              AXI_LEDS = ~AXI_SWITCHES;
92
93              // Les afficheurs HEX5 à HEX0 affichent en hexadécimal l'inverse des bits
                23 à 0 de la
94              // constante définie dans l'IP.
95              AXI_HEX0 = ~tab_dec_to_hex_7seg[cst & 0xF];
96              AXI_HEX1 = ~tab_dec_to_hex_7seg[(cst>>4)  & 0xF];
97              AXI_HEX2 = ~tab_dec_to_hex_7seg[(cst>>8)  & 0xF];
98              AXI_HEX3 = ~tab_dec_to_hex_7seg[(cst>>12) & 0xF];
99              AXI_HEX4 = ~tab_dec_to_hex_7seg[(cst>>16) & 0xF];
100             AXI_HEX5 = ~tab_dec_to_hex_7seg[(cst>>20) & 0xF];
101
102         // Si le bouton 2 est pressé (via une interruption)
103         } else if (irqKey2) {
104             irqKey2 = 0;
105
106             /*  l'affichage des LEDs et des afficheurs 7 segments subit unerotation à
                droite */
107             led_tmp = AXI_LEDS & 0x1;
108             AXI_LEDS = ((AXI_LEDS & 0x3ff) >> 1) | (led_tmp << 9);
109
110             Seg_tmp = AXI_HEX0;
111             AXI_HEX0 = AXI_HEX1;
112             AXI_HEX1 = AXI_HEX2;
113             AXI_HEX2 = AXI_HEX3;
114             AXI_HEX3 = AXI_HEX4;
115             AXI_HEX4 = AXI_HEX5;
116             AXI_HEX5 = Seg_tmp;
117
118
119         // Si le bouton 3 est pressé (via une interruption)
120         } else if (irqKey3) {
121             irqKey3 = 0;
122
123             /* l'affichage des LEDs et des afficheurs 7 segments subit une rotation à
```

```
                gauche */
124             led_tmp = AXI_LEDS & 0x200;
125             AXI_LEDS = (AXI_LEDS << 1) | (led_tmp >> 9);
126
127             Seg_tmp = AXI_HEX5;
128             AXI_HEX5 = AXI_HEX4;
129             AXI_HEX4 = AXI_HEX3;
130             AXI_HEX3 = AXI_HEX2;
131             AXI_HEX2 = AXI_HEX1;
132             AXI_HEX1 = AXI_HEX0;
133             AXI_HEX0 = Seg_tmp;
134
135         }
136
137     }
138
139 }
140
141 /* Routine d'interruption */
142 void pushbutton_ISR(void){
143     // Permet de tester le masquage
144     // static int cpt_int = 0;
145
146     /* Lecture et acquitement des interruptions */
147     int src_irq = AXI_INT_SRC;
148
149     // Key2 pressé
150     if (src_irq & KEY2) {
151         irqKey2 = 1;
152     }
153
154     // Key3 pressé
155     if (src_irq & KEY3) {
156         irqKey3 = 1;
157     }
158
159
160     // Tous les 3 interruptions de KEY0 et KEY1, change le masque de key 2 et 3
161     /*
162     if (src_irq & KEY0 || src_irq & KEY1) {
163         cpt_int++;
164
165         if (cpt_int % 3 == 0)
166             AXI_INT_MASK = AXI_INT_MASK ^ (KEY3 | KEY2);
167     }
168     */
169
170 }
171
```