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

#include "defines.h"


/* Variable globales */

int irqKey2 = 0;
int irqKey3 = 0;




int main(void){

    // tableau de conversion         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      // Masque le bouton key3 (pour tester le masquage des interruptions)
59      // AXI_INT_MASK = KEY3;
60
61      disable_A9_interrupts();   // disable interrupts in the A9 processor
62      set_A9_IRQ_stack();        // initialize the stack pointer for IRQ mode
63      config_GIC();              // configure the general interrupt controller
64      config_KEYs();             // configure KEYs to generate interrupts
65      enable_A9_interrupts();    // enable interrupts in the A9 processor
66
67
68
69
70      while(1){
71          /* Appuie sur KEY 0*/
72          if ((AXI_KEYS & KEY0) == 0) {
73              // l'états des switches est copiés sur les LEDs.
74              AXI_LEDS = AXI_SWITCHES;
75              // Les afficheurs HEX5 à HEX0 affichent en hexadécimal les bits 23 à 0 de
                la constante définie dans l'IP.
76              AXI_HEX0 = tab_dec_to_hex_7seg[cst & 0xF];
77              AXI_HEX1 = tab_dec_to_hex_7seg[(cst>>4)  & 0xF];
78              AXI_HEX2 = tab_dec_to_hex_7seg[(cst>>8)  & 0xF];
79              AXI_HEX3 = tab_dec_to_hex_7seg[(cst>>12) & 0xF];
80              AXI_HEX4 = tab_dec_to_hex_7seg[(cst>>16) & 0xF];
81              AXI_HEX5 = tab_dec_to_hex_7seg[(cst>>20) & 0xF];
82
83
84          /* Appuie sur KEY 1 */
85          } else if ((AXI_KEYS & KEY1) == 0) {
86              //  l'états inverses des switches est copiés sur les LEDs.
87              AXI_LEDS = ~AXI_SWITCHES;
88
89              // Les afficheurs HEX5 à HEX0 affichent en hexadécimal l'inverse des bits
                23 à 0 de la
90              // constante définie dans l'IP.
91              AXI_HEX0 = ~tab_dec_to_hex_7seg[cst & 0xF];
92              AXI_HEX1 = ~tab_dec_to_hex_7seg[(cst>>4)  & 0xF];
93              AXI_HEX2 = ~tab_dec_to_hex_7seg[(cst>>8)  & 0xF];
94              AXI_HEX3 = ~tab_dec_to_hex_7seg[(cst>>12) & 0xF];
95              AXI_HEX4 = ~tab_dec_to_hex_7seg[(cst>>16) & 0xF];
96              AXI_HEX5 = ~tab_dec_to_hex_7seg[(cst>>20) & 0xF];
97
98          // Si le bouton 2 est pressé (via une interruption)
99          } else if (irqKey2) {
100             irqKey2 = 0;
101
102             /*  l'affichage des LEDs et des afficheurs 7 segments subit unerotation à
                droite */
103             led_tmp = AXI_LEDS & 0x1;
104             AXI_LEDS = ((AXI_LEDS & 0x3ff) >> 1) | (led_tmp << 9);
105
106             Seg_tmp = AXI_HEX0;
107             AXI_HEX0 = AXI_HEX1;
108             AXI_HEX1 = AXI_HEX2;
109             AXI_HEX2 = AXI_HEX3;
110             AXI_HEX3 = AXI_HEX4;
111             AXI_HEX4 = AXI_HEX5;
112             AXI_HEX5 = Seg_tmp;
113
114
115         // Si le bouton 3 est pressé (via une interruption)
116         } else if (irqKey3) {
117             irqKey3 = 0;
118
119             /* l'affichage des LEDs et des afficheurs 7 segments subit une rotation à
                gauche */
120             led_tmp = AXI_LEDS & 0x200;
121             AXI_LEDS = (AXI_LEDS << 1) | (led_tmp >> 9);
122
```

```
123                 Seg_tmp = AXI_HEX5;
124                 AXI_HEX5 = AXI_HEX4;
125                 AXI_HEX4 = AXI_HEX3;
126                 AXI_HEX3 = AXI_HEX2;
127                 AXI_HEX2 = AXI_HEX1;
128                 AXI_HEX1 = AXI_HEX0;
129                 AXI_HEX0 = Seg_tmp;
130 
131 
132             }
133 
134             AXI_HEX5 = test1;
135         }
136         AXI_HEX5 = test1;
137 
138     }
139 
140     /* Routine d'interruption */
141     void pushbutton_ISR(void){
142         // Permet de tester le masquage
143         // static int cpt_int = 0;
144 
145         /* Lecture et acquitement des interruptions */
146         int src_irq = AXI_INT_SRC;
147 
148         // Key2 pressé
149         if (src_irq & KEY2) {
150             irqKey2 = 1;
151         }
152 
153         // Key3 pressé
154         if (src_irq & KEY3) {
155             irqKey3 = 1;
156         }
157 
158 
159         // Tous les 3 interruptions de KEY0 et KEY1, change le masque de key 2 et 3
160         /*
161         if (src_irq & KEY0 || src_irq & KEY1) {
162             cpt_int++;
163 
164             if (cpt_int % 3 == 0)
165                 AXI_INT_MASK = AXI_INT_MASK ^ (KEY3 | KEY2);
166         }
167         */
168 
169     }
170 
```