ARC2 – TP 8

Prune Forget, Léo Noël-Baron & Thierry Sampaio

23/03/2016

Ce TP propose d'utiliser les mécanismes d'interruption du Nios pour gérer des événements matériels (ici une souris et une animation à l'écran).

Activation des interruptions

On commence par implémenter les fonctions d'activation :

```
1 void activer_interruptions() {
2     unsigned int tmp;
3     NIOS2_READ_STATUS(tmp);
4     tmp |= 1; // Mettre le bit 0 de STATUS à 1
5     NIOS2_WRITE_STATUS(tmp);
6 }
```

```
1 void activer_interruption(int num) {
2     unsigned int tmp;
3     NIOS2_READ_IENABLE(tmp);
4     tmp |= (1 << num); // Mettre le bit num de IENABLE à 1
5     NIOS2_WRITE_IENABLE(tmp);
6 }
```

et on appelle la première au début du `main`.

Contrôle de l'animation

Pour gérer l'animation à l'écran, on doit lancer le timer en mode continu soit, d'après l'énoncé, `0111` ce qui donne `*TIMER_CTRL = 7` dans le `main`. On doit ensuite appeler `activer_interruption(INTERVAL_TIMER_IRQ)` pour activer l'interruption du timer. Enfin, il faut implémenter la routine d'interruption :

```
1 static void TIMER_ISR(void *context, alt_u32 id) {
2     tick();
3     *TIMER_STATUS = 0;
4 }
```

et ô miracle, les nuages défilent.

Contrôle de la souris

Pour gérer la souris, il faut d'abord activer l'interruption correspondante dans le `main` puis écrire la routine suivante :

```
1 char pnext = 0, p[3];
2 static void MOUSE_ISR(void *context, alt_u32 id) {
3     p[pnext] = (*PS2_DATA) & 0xFF; // Récupérer un octet de données
4     if(++pnext == 3) { // Si on a eu les deux déplacements
5         x_pos = MOD(x_pos + p[1], 320);
6         y_pos = MOD(y_pos - p[2], 240);
7         pnext = 0; // RAZ
8     }
9 }
```

qui lit les octets de données de la souris un à un et met à jour la position du curseur quand il le faut. Et ô joie, le canari vole.

main.c

```c
#include "nios2.h"
#include "system.h"
#include "sys/alt_irq.h"
#include "init.h"

#include "graphlib.h"
#include "cursor.h"
#include "nuages.h"

#define MOD(a,b) ((a)%(b)+(b))%(b)

volatile unsigned int* TIMER_STATUS = (unsigned int *) INTERVAL_TIMER_BASE;
volatile unsigned int* TIMER_CTRL = (unsigned int *) (INTERVAL_TIMER_BASE + 4);
volatile unsigned int* PS2_DATA = (unsigned int *) PS2_PORT_BASE;

// Question 1.1
void activer_interruptions() {
    unsigned int tmp;
    NIOS2_READ_STATUS(tmp);
    tmp |= 1;
    NIOS2_WRITE_STATUS(tmp);
}

// Question 1.2
void activer_interruption(int num) {
    unsigned int tmp;
    NIOS2_READ_IENABLE(tmp);
    tmp |= (1 << num);
    NIOS2_WRITE_IENABLE(tmp);
}

int x_pos = 100, y_pos = 100;
char pnext = 0, p[3];
int middle_pos = 0;

void tick() {
    middle_pos = MOD(middle_pos - 1, 320);
}

// Question 3
static void TIMER_ISR(void *context, alt_u32 id) {
    tick();
    *TIMER_STATUS = 0;
}

```

```
46 // Question 4.2
47 static void MOUSE_ISR(void *context, alt_u32 id) {
48     p[pnext] = (*PS2_DATA) & 0xFF;
49     if(++pnext == 3) {
50         x_pos = MOD(x_pos + p[1], 320);
51         y_pos = MOD(y_pos - p[2], 240);
52         pnext = 0;
53     }
54 }
55
56 int main() {
57     init();
58
59     // Question 1.3
60     activer_interruptions();
61
62     // Question 2.2
63     *TIMER_CTRL = 7;
64
65     // Question 2.3
66     activer_interruption(INTERVAL_TIMER_IRQ);
67
68     // Question 4.1
69     activer_interruption(PS2_PORT_IRQ);
70
71     // Enregistrement des routines d'interruption.
72     alt_irq_register ( INTERVAL_TIMER_IRQ, 0, TIMER_ISR );
73     alt_irq_register( PS2_PORT_IRQ, 0, MOUSE_ISR );
74
75     while(1) {
76         clear_screen();
77         draw_image((unsigned short *)nuages_img, 320, 240, middle_pos-320, 0);
78         draw_image((unsigned short *)nuages_img, 320, 240, middle_pos, 0);
79         draw_piou(x_pos, y_pos);
80         swap_buffers();
81     }
82 }
```