LINGI 2315 - Homework 2 - My Bare-metal App **P1**

Name	Delcoigne Ben	Noma	38771700

Description of the software that implements the control of the blinking frequency of LED[0] with relevant screenshots of your code

The software implements are the same as last homework,except now I’m using interrupts to update the frequency instead of in every while loop.

```
// Alive LED0

reg [25: 0] counter;
reg led_level;
always @(posedge fpga_clk_50) begin
    if (counter >= fpga_pio0) begin //On change le 24999999 par le pio qu'on a ajouté de 26 bits
        counter <= 0;
        led_level <= ~led_level;
    end
    else
        counter <= counter + 1'b1;
end

assign LED[0] = led_level;


endmodule
```

Here is the same function as last time:

```
void handle_fpga_buttons(){
    //I just added the code of last homework.
    uint32_t button = alt_read_word(fpga_buttons);
    uint32_t frequency = alt_read_word(fpga_pio_0);

    if (button == 1){
        frequency  += PIO_0_FREQ/10;
        alt_write_word(fpga_pio_0, frequency);
    }
    if (button == 2){
        frequency -= PIO_0_FREQ/10;
        alt_write_word(fpga_pio_0, frequency);
    }
}
```

```
/* Interrupt service routine for the buttons */
void button_isr_callback (uint32_t icciar, void *context)
{
    // Do something
    printf("INFO: IRQ from buttons : %x (edge = %x)\r\n", alt_read_word(fpga_buttons), alt_read_word(fpga_buttons + 12));

    //I here add the call to handle what happens when a button was pressed --> code from last homework
    //This was suggested to me by a friend, I would just have put it on the main loop otherwise.
    handle_fpga_buttons();
    // Clear the interrupt
    alt_gpt_int_clear_pending (GPT_BUTTON_IRQ);
    // Clear the interruptmask of PIO core
    alt_write_word(fpga_buttons + (2*4), 0x0);

    // Enable the interruptmask and edge register of PIO core for new interrupt
    alt_write_word(fpga_buttons + (2*4), 0x3);
    alt_write_word(fpga_buttons + (3*4), 0x3);
}
```

LINGI 2315 - Homework 2 - My Bare-metal App **P2**

Name		Noma	

Description of the software the implements the display of 1-axis acceleration with relevant screenshots of your code

I used the same code as given for the button for the interrupts:

```
void sw_isr_callback (uint32_t icciar, void *context)
{
    // Do something
    printf("INFO: IRQ from switchh");

    //I here add the call to handle what happens when a button was pressed --> code from last homework
    //This was suggested to me by a friend, I would just have put it on the main loop otherwise.
    read_switches();
    // Clear the interrupt
    alt_gpt_int_clear_pending (GPT_SW_IRQ);
    // Clear the interruptmask of PIO core
    alt_write_word(fpga_switches + (2*4), 0x0);

    // Enable the interruptmask and edge register of PIO core for new interrupt
    alt_write_word(fpga_switches + (2*4), 0xf);
    alt_write_word(fpga_switches + (3*4), 0xf);
}
```

```
//SWITCH INTERRUPT
assert(ALT_E_SUCCESS == alt_int_isr_register(GPT_SW_IRQ, sw_isr_callback , NULL));

// Ignore target_set() for non-SPI interrupts
if ((ALT_INT_INTERRUPT_F2S_FPGA_IRQ0 + DIPSW_PIO_IRQ) >= 32) {
    // Set timer interrupt distributor target (=0x3) (1 = CPU0, 2=CPU1)
    assert(ALT_E_SUCCESS == alt_int_dist_target_set (GPT_SW_IRQ, (1 << alt_int_util_cpu_count()) - 1));
}
// Enable distributor interrupt
assert(ALT_E_SUCCESS == alt_int_dist_enable(GPT_SW_IRQ));
alt_write_word(fpga_switches + (2*4), 0xf);
alt_write_word(fpga_switches + (3*4), 0xf);
```

When the interrupt is enabled, i modify a global variable (which stores which switch was on → the axis we're reading)

```
void read_switches(){
    uint32_t switchreading = alt_read_word(fpga_switches);

    int switchvalue = (int)(switchreading) % 4;
    if (switchvalue == 0){
        switchvalue = 1;
    }
    current_switch = switchvalue;
```

In the main loop, i every time handle the LEDS

```
int main() {
    printf("\r\nDE10-Nano - MyApp_Gsensor\r\n\r\n");

    setup_hps_interrupt();

    setup_hps_timer();
    setup_hps_gpio();
    setup_hps_i2c();
    setup_fpga_leds();
    alt_write_word(fpga_pio_0, 24999999);//Init the default value

    while (true) {
        handle_hps_led();
        //handle_fpga_buttons(); //Removed because is not done often and added in the interruption code.
        //handle_fpga_leds();
      //  read_switches();
        int toprint = handle_gsensor(current_switch);
        print_on_leds(toprint);
        printf("%d", current_switch);
        //handle_gsensor(1);
        delay_us(ALT_MICROSECS_IN_A_SEC/100);
    }

    return 0;
}
```

Handle Gsensor reads the value of the gsensor on the selected axis, and then returns the read value

```
72
73  int handle_gsensor(int axis) {
74      bool bSuccess;
75      const int mg_per_digi = 4;
76      uint16_t szXYZ[3];
77
78      if (ADXL345_IsDataReady(&i2c_dev)){
79          bSuccess = ADXL345_XYZ_Read(&i2c_dev, szXYZ);
80          if (bSuccess){
81              printf("X=%d mg, Y=%d mg, Z=%d mg\r\n",(int16_t)szXYZ[0]*mg_per_digi, (int16_t)szXYZ[1]*mg_per_digi, (int16_t)szXYZ[2]*mg_per_digi);
82              // show raw data,
83              //printf("X=%04x, Y=%04x, Z=%04x\r\n", (alt_u16)szXYZ[0], (alt_u16)szXYZ[1],(alt_u16)szXYZ[2]);
84
85              axis-=1;
86              return (int)szXYZ[axis]*mg_per_digi;
87          }
88          else return 0;
89      }
90      return 0;
91  }
```

This value is then used to print using print_on_leds();

```
}
void print_on_leds(int sensoroutput){

    int32_t led_mask=alt_read_word(fpga_leds);
    int16_t reading = sensoroutput;

    if(reading <=0) reading = -reading;
    reading=(int16_t)reading/300;

    switch(reading){

    case 0:
        led_mask = 0x0;
        break;
    case 1:
        led_mask = 0x1;
        break;
    case 2:
        led_mask = 0x3;
        break;
    case 3:
        led_mask = 0x7;
        break;
    case 4:
        led_mask = 0xf;
        break;
    case 5:
        led_mask = 0x1f;
        break;
    case 6:
        led_mask = 0x3f;
        break;
    case 7:
        led_mask = 0x7f;
        break;

    default:
        break;


    }
    alt_write_word(fpga_leds, led_mask);


}
```