```c
//Examen practico Ejercicio 1
//Antonio Corona
//Bernardo Urriza
//Fernando Cossio
#include <asf.h>

#define BTN_UP   AVR32_PIN_PB22
#define BTN_DOWN AVR32_PIN_PB23
#define BTN_RIGHT AVR32_PIN_PB24
#define BTN_LEFT AVR32_PIN_PB25
#define BTN_CENTER AVR32_PIN_PB26

#define LEDO   AVR32_PIN_PB27
#define LED1   AVR32_PIN_PB28
#define LED2   AVR32_PIN_PA05
#define LED3   AVR32_PIN_PA06

enum btn{NONE, UP, DOWN, LEFT, RIGHT, CENTER};
enum btn btn_pressed = NONE;
uint8_t state = 0;

uint8_t counter =0;



__attribute__ ((__interrupt__));
void Botones (void);

//Init FN y Variables Globales
void inicializa_PM(void);
void Inicializa_PLL(uint8_t mul);
void Prender_Leds(uint8_t value);
void state0(void);
void state1(void);
void state2(void);

int main (void)
{

    inicializa_PM();
    delay_init(12000000);

    board_init();

    Disable_global_interrupt();
    INTC_init_interrupts();
    INTC_register_interrupt(&Botones, 70, 3);
    INTC_register_interrupt(&Botones, 71, 3);

    uint16_t button_ref [] = {BTN_UP,BTN_DOWN,BTN_RIGHT,BTN_LEFT,BTN_CENTER};
    for(uint8_t i=0; i<5; i++){
        gpio_enable_gpio_pin(button_ref[i]);
        gpio_enable_pin_pull_up(button_ref[i]);
        gpio_enable_pin_interrupt(button_ref[i],GPIO_FALLING_EDGE);
    }

    Enable_global_interrupt();

    Prender_Leds(0b000);//apagar leds

    while (true)
    {
        switch (state) {
            case 0: //contador arriba y abajo
                state0();
                break;
            case 1: //
                state1();
                break;
            case 2:
                state2();
                break;
        } //Fin switch
    } //Fin While
}//Fin de Main

void state0(void){
    while(state==0){
        if (btn_pressed==UP && counter < 15){
            counter ++;
            Prender_Leds(counter);
            btn_pressed=NONE; //IRQ atendida
        }else if(btn_pressed==DOWN && counter > 0){
            counter --;
            Prender_Leds(counter);
            btn_pressed=NONE; //IRQ atendida
        }else if(btn_pressed==CENTER){
            counter = 0;
            Prender_Leds(counter);
            btn_pressed=NONE; //IRQ atendida
        }
    }
}
void state1(void){
    uint8_t numero = 0b0001;//Este numero en bin: 1000, 0100, 0010, 0001 (8,4,2,1)
    uint8_t mul = 3; //Para PLL0
    while(state==1){
            if (btn_pressed != CENTER){
                if (numero == 1){
                    mul = (mul+1)%4;
                    Inicializa_PLL(mul+3);
                    numero =0b1000;
                }else{
                        numero = numero >> 1; //
                }
            }
            for (U32 i = 0; i<100000; i++){
                Prender_Leds(numero);
            }
    }
}
void state2(void){
    uint8_t numero = 0b1000;
    uint8_t mul = 3; //Para PLL0
    while(state==2){
        if (btn_pressed != CENTER){
            if (numero == 0b1000){
                mul = (mul+1)%4;
                Inicializa_PLL(mul+3);
                numero = 0b0001;
            }else{
                    numero = numero << 1;
            }
        }for (U32 i = 0; i<100000; i++){
            Prender_Leds(~numero);
        }
    }
}


void Prender_Leds(uint8_t value){
    if ((value & 0b1000)>>3)gpio_clr_gpio_pin(LED0); else gpio_set_gpio_pin(LED0);
    if ((value & 0b0100)>>2)gpio_clr_gpio_pin(LED1); else gpio_set_gpio_pin(LED1);
    if ((value & 0b0010)>>1)gpio_clr_gpio_pin(LED2); else gpio_set_gpio_pin(LED2);
    if (value & 0b0001 )    gpio_clr_gpio_pin(LED3); else gpio_set_gpio_pin(LED3);
}//Fin Fn

void Botones (void){
    if (gpio_get_pin_interrupt_flag(BTN_UP)) {
        btn_pressed=UP;
        state=0;
        gpio_clear_pin_interrupt_flag(BTN_UP);
    }
    if (gpio_get_pin_interrupt_flag(BTN_DOWN)){
        btn_pressed=DOWN;
        state=0;
        gpio_clear_pin_interrupt_flag(BTN_DOWN);
    }
    if (gpio_get_pin_interrupt_flag(BTN_RIGHT)){
        btn_pressed=RIGHT;
        state=1;
        gpio_clear_pin_interrupt_flag(BTN_RIGHT);
    }
    if (gpio_get_pin_interrupt_flag(BTN_LEFT)){
        btn_pressed=LEFT;
        state=2;
        gpio_clear_pin_interrupt_flag(BTN_LEFT);
    }
    if (gpio_get_pin_interrupt_flag(BTN_CENTER)){
        gpio_clear_pin_interrupt_flag(BTN_CENTER);
        btn_pressed=CENTER;
        }
    if (gpio_get_pin_interrupt_flag(BTN_CENTER)){
        gpio_clear_pin_interrupt_flag(BTN_CENTER);
    }
} //Fin Botones

void inicializa_PM (void){
    pm_switch_to_osc0(&AVR32_PM,12000000,3); //fOSC= 12MHz, startup 18ms
    flashc_set_wait_state(1);
} //Fin PM

void Inicializa_PLL(uint8_t mul){
    pm_switch_to_osc0(&AVR32_PM, 12000000,3);
    pm_pll_disable(&AVR32_PM,0);
    pm_pll_setup(&AVR32_PM,0, mul, 1,0,16); //pll0, mul variable, div = 1, 16 lockcount
    pm_pll_set_option(&AVR32_PM,0,1,0,0);  //pll0, 80-180, no divide/2, start normal
    pm_pll_enable(&AVR32_PM,0);
    pm_wait_for_pll0_locked(&AVR32_PM);
    flashc_set_wait_state(1);
    pm_switch_to_clock(&AVR32_PM,2);//PLL como MC
}//Fin Fn

//PARA FOSC=12 MHz
//mul=3 fpll=96MHz
//mul=4 fpll=120MHz
//mul=5 fpll=144MHz
//mul=6 fpll=168MHz

```