

```

1 //Examen practico Ejercicio 1
2 //Antonio Corona
3 //Bernardo Urriza
4 //Fernando Cossio
5 #include <asf.h>
6
7 #define BTN_UP    AVR32_PIN_PB22
8 #define BTN_DOWN  AVR32_PIN_PB23
9 #define BTN_RIGHT AVR32_PIN_PB24
10 #define BTN_LEFT  AVR32_PIN_PB25
11 #define BTN_CENTER AVR32_PIN_PB26
12
13 #define LED0    AVR32_PIN_PB27
14 #define LED1    AVR32_PIN_PB28
15 #define LED2    AVR32_PIN_PA05
16 #define LED3    AVR32_PIN_PA06
17
18 enum btn{NONE, UP, DOWN, LEFT, RIGHT, CENTER};
19 enum btn btn_pressed = NONE;
20 uint8_t state = 0;
21
22 uint8_t counter =0;
23
24
25
26 __attribute__ ((__interrupt__));
27 void Botones (void);
28
29 //Init FN y Variables Globales
30 void inicializa_PM(void);
31 void Inicializa_PLL(uint8_t mul);
32 void Prender_Leds(uint8_t value);
33 void state0(void);
34 void state1(void);
35 void state2(void);
36
37 int main (void)
38 {
39
40     inicializa_PM();
41     delay_init(12000000);
42
43     board_init();
44
45     Disable_global_interrupt();
46     INTC_init_interrupts();
47     INTC_register_interrupt(&Botones, 70, 3);
48     INTC_register_interrupt(&Botones, 71, 3);
49
50     uint16_t button_ref [] = {BTN_UP,BTN_DOWN,BTN_RIGHT,BTN_LEFT,BTN_CENTER};
51     for(uint8_t i=0; i<5; i++){
52         gpio_enable_gpio_pin(button_ref[i]);
53         gpio_enable_pin_pull_up(button_ref[i]);
54         gpio_enable_pin_interrupt(button_ref[i],GPIO_FALLING_EDGE);
55     }
56

```

```
57     Enable_global_interrupt();
58
59     Prender_Leds(0b000); //apagar leds
60
61     while (true)
62     {
63         switch (state) {
64             case 0: //contador arriba y abajo
65                 state0();
66                 break;
67             case 1: //
68                 state1();
69                 break;
70             case 2:
71                 state2();
72                 break;
73         } //Fin switch
74     } //Fin While
75 } //Fin de Main
76
77 void state0(void){
78     while(state==0){
79         if (btn_pressed==UP && counter < 15){
80             counter ++;
81             Prender_Leds(counter);
82             btn_pressed=NONE; //IRQ atendida
83         }else if(btn_pressed==DOWN && counter > 0){
84             counter --;
85             Prender_Leds(counter);
86             btn_pressed=NONE; //IRQ atendida
87         }else if(btn_pressed==CENTER){
88             counter = 0;
89             Prender_Leds(counter);
90             btn_pressed=NONE; //IRQ atendida
91         }
92     }
93 }
94 void state1(void){
95     uint8_t numero = 0b0001; //Este numero en bin: 1000, 0100, 0010, 0001 (8,4,2,1)
96     uint8_t mul = 3; //Para PLL0
97     while(state==1){
98         if (btn_pressed != CENTER){
99             if (numero == 1){
100                 mul = (mul+1)%4;
101                 Inicializa_PLL(mul+3);
102                 numero = 0b1000;
103             }else{
104                 numero = numero >> 1; //
105             }
106         }
107         for (U32 i = 0; i<100000; i++){
108             Prender_Leds(numero);
109         }
110     }
111 }
112 void state2(void){
```

```
113     uint8_t numero = 0b1000;
114     uint8_t mul = 3; //Para PLL0
115     while(state==2){
116         if (btn_pressed != CENTER){
117             if (numero == 0b1000){
118                 mul = (mul+1)%4;
119                 Inicializa_PLL(mul+3);
120                 numero = 0b0001;
121             }else{
122                 numero = numero << 1;
123             }
124         }for (U32 i = 0; i<100000; i++){
125             Prender_Leds(~numero);
126         }
127     }
128 }
129
130
131 void Prender_Leds(uint8_t value){
132     if ((value & 0b1000)>>3)gpio_clr_gpio_pin(LED0); else gpio_set_gpio_pin(LED0);
133     if ((value & 0b0100)>>2)gpio_clr_gpio_pin(LED1); else gpio_set_gpio_pin(LED1);
134     if ((value & 0b0010)>>1)gpio_clr_gpio_pin(LED2); else gpio_set_gpio_pin(LED2);
135     if (value & 0b0001 )    gpio_clr_gpio_pin(LED3); else gpio_set_gpio_pin(LED3);
136 }//Fin Fn
137
138 void Botones (void){
139     if (gpio_get_pin_interrupt_flag(BTN_UP)) {
140         btn_pressed=UP;
141         state=0;
142         gpio_clear_pin_interrupt_flag(BTN_UP);
143     }
144     if (gpio_get_pin_interrupt_flag(BTN_DOWN)){
145         btn_pressed=DOWN;
146         state=0;
147         gpio_clear_pin_interrupt_flag(BTN_DOWN);
148     }
149     if (gpio_get_pin_interrupt_flag(BTN_RIGHT)){
150         btn_pressed=RIGHT;
151         state=1;
152         gpio_clear_pin_interrupt_flag(BTN_RIGHT);
153     }
154     if (gpio_get_pin_interrupt_flag(BTN_LEFT)){
155         btn_pressed=LEFT;
156         state=2;
157         gpio_clear_pin_interrupt_flag(BTN_LEFT);
158     }
159     if (gpio_get_pin_interrupt_flag(BTN_CENTER)){
160         gpio_clear_pin_interrupt_flag(BTN_CENTER);
161         btn_pressed=CENTER;
162     }
163     if (gpio_get_pin_interrupt_flag(BTN_CENTER)){
164         gpio_clear_pin_interrupt_flag(BTN_CENTER);
165     }
166 } //Fin Botones
167
168 void inicializa_PM (void){
```

```
169     pm_switch_to_osc0(&AVR32_PM,12000000,3); //fOSC= 12MHz, startup 18ms
170     flashc_set_wait_state(1);
171 } //Fin PM
172
173 void Inicializa_PLL(uint8_t mul){
174     pm_switch_to_osc0(&AVR32_PM, 12000000,3);
175     pm_pll_disable(&AVR32_PM,0);
176     pm_pll_setup(&AVR32_PM,0, mul, 1,0,16); //pll0, mul variable, div = 1, 16 lockcount
177     pm_pll_set_option(&AVR32_PM,0,1,0,0); //pll0, 80-180, no divide/2, start normal
178     pm_pll_enable(&AVR32_PM,0);
179     pm_wait_for_pll0_locked(&AVR32_PM);
180     flashc_set_wait_state(1);
181     pm_switch_to_clock(&AVR32_PM,2); //PLL como MC
182 } //Fin Fn
183
184 //PARA FOSC=12 MHz
185 //mul=3 fpll=96MHz
186 //mul=4 fpll=120MHz
187 //mul=5 fpll=144MHz
188 //mul=6 fpll=168MHz
189
```