```c
//Actividad Práctica Timer  14 octubre 2019

//Bernardo Urriza A01336299
//Antonio Corona A01337294
//Fernando Cossio A00759499

#include <avr32/io.h>
#include "compiler.h"
#include "power_clocks_lib.h"
#include "board.h"
#include "gpio.h"
#include "pwm.h"
#include "tc.h"
#include "pm.h"
#include "intc.h"

#define BTN_RIGHT AVR32_PIN_PB24
#define BTN_CENTER AVR32_PIN_PB26

//Funciones de usuario
__attribute__ ((__interrupt__))
void teclas(void);
static void init_tc_input(volatile avr32_tc_t *tc, unsigned int channel);
static void init_tc_output(volatile avr32_tc_t *tc, unsigned int channel);

//Variables globales
uint8_t state=0;
int ra_input = 0, ra_output = 0;

int main (void){

    // Configuracion de PM
    pm_switch_to_osc0(&AVR32_PM, 16000000, 3); //F pba= 16MHz; startup: 18ms
    //Inicializacion de canales de captura y PWM
    init_tc_input(&AVR32_TC, 0); //Canal 0 como captura
    init_tc_output(&AVR32_TC, 2); //Canal 2 como waveform
    //Inicializacion de GPIO por IRQ
    static const gpio_map_t TC_GPIO_MAP =
    {
        {106, 2}, //GPIO 106, TIOA0, FN especial C, 2
        {86, 2} //GPIO 86, FN especial C, 2
    };
    gpio_enable_module(TC_GPIO_MAP, sizeof(TC_GPIO_MAP) / sizeof(TC_GPIO_MAP[0]));//Activar Fn
      especiales para TIOA0 y TIOA2
    gpio_enable_gpio_pin(BTN_RIGHT);
    gpio_enable_gpio_pin(BTN_CENTER);

    //Interrupciones
    Disable_global_interrupt();
    INTC_init_interrupts();
    INTC_register_interrupt(&teclas, 71, 3); //Hab interrupcion para teclas RIGHT Y CENTER,
    gpio_enable_pin_interrupt(BTN_RIGHT , GPIO_FALLING_EDGE);
    gpio_enable_pin_interrupt(BTN_CENTER , GPIO_FALLING_EDGE);
    Enable_global_interrupt();

    //fPBA=16MHz; fPBA/32=500kHz => TPBA=2us
```

```
56      //Tpwm= 30ms =>  rc = Tpwm/TPBA = 15,000
57      tc_write_rc(&AVR32_TC0, 2, 15000); //Periodo 30ms en pwm.
58      tc_write_ra(&AVR32_TC0, 2, 7500); //Valor por defecto de pwm al 50% Duty
59      gpio_set_gpio_pin(86); //Para iniciar PWM en 1
60
61      while (1) {
62          switch(state){
63              case 0: //Int tecla center
64              // deshabilitar timers
65              tc_stop(&AVR32_TC0,0);
66              tc_stop(&AVR32_TC0,2);
67              break;
68              case 1: //Se presiona tecla Right Comienza a capturar
69              tc_start(&AVR32_TC0,0); //Inicia Captura
70              tc_start(&AVR32_TC0,2); //Inicia PWM, trigger por SW
71              while(state == 1){ //evita que se repita el tc_start
72                  ra_input = tc_read_ra(tc, 0); //Inicia con este valor el PWM
73                  if (ra_input < 4500){ //30%(15000) = 4500
74                      //generar pwm con T=30ms y DC=20%
75                      //raPWM = rc * 0.2 = 3000
76                      ra_output = 3000;
77                      tc_write_ra(&AVR32_TC0, 2, ra_output);
78                      }else if(ra_input <= 10500){ //70%(15000) = 10500
79                      //generar pwm con T=30ms y DC=50%
80                      //raPWM = rc * 0.5 = 7500
81                      ra_output = 7500;
82                      tc_write_ra(&AVR32_TC0, 2, ra_output);
83                      }else{
84                      //generar pwm con T=30ms y DC=80%
85                      //raPWM = rc * 0.8 = 12000
86                      ra_output = 12000;
87                      tc_write_ra(&AVR32_TC0, 2, ra_output);
88                  }
89              }
90              break;
91          }//Fin switch
92      }//Fin While
93  } //Fin Main
94
95  //Handler
96  void teclas (void) {
97      if (gpio_get_pin_interrupt_flag (BTN_CENTER)){
98          state = 0;
99          gpio_clear_pin_interrupt_flag(BTN_CENTER);
100     }
101     if (gpio_get_pin_interrupt_flag (BTN_RIGHT)){
102         state = 1;
103         gpio_clear_pin_interrupt_flag(BTN_RIGHT);
104     }
105     gpio_get_pin_interrupt_flag (BTN_RIGHT);//Para que funcione en EVK1105
106 }//Fin Teclas
107
108 static void init_tc_input(volatile avr32_tc_t *tc, unsigned int channel){//Para captura, carga ↵
      en RA el tiempo en alto
109     // Options for capture mode.
110     tc_capture_opt_t capture_opt =
```

```
111     {
112         .channel  = channel, //Canal
113 
114         .ldrb     = 0, //No hay carga en TC_SEL_NO_EDGE,
115         .ldra     = 2, //Carga en Falling de la entrada TIOA TC_SEL_FALLING_EDGE,
116 
117         .cpctrg   = 0, //Compare con RC no detiene la captura TC_NO_TRIGGER_COMPARE_RC
118         .abetrg   = 1, //Trigger por la misma TIOA TC_EXT_TRIG_SEL_TIOA
119         .etrgedg  = 1, //Rising es trigger, TC_SEL_RISING_EDGE
120 
121         .ldbdis   = FALSE, //Se va a medir mas de un periodo
122         .ldbstop  = FALSE, //Se va a medir mas de un periodo
123 
124         .burst    = 0, //Sin Burst, TC_BURST_NOT_GATED
125         .clki     = 0, //Reloj no invertido, TC_CLOCK_RISING_EDGE
126         .tcclks   = 3, // fPBA/32, TC4, TC_CLOCK_SOURCE_TC4
127     };
128 
129     // Initialize the timer/counter capture.
130     tc_init_capture(tc, &capture_opt);
131 }//init_tc_input
132 
133 
134 static void init_tc_output(volatile avr32_tc_t *tc, unsigned int channel){
135     // Options for waveform generation.
136     tc_waveform_opt_t waveform_opt =
137     {
138         .channel  = channel,                                        // Channel selection.
139 
140         .bswtrg   = 0, //TC_EVT_EFFECT_NOOP,           // Software trigger effect on TIOB.
141         .beevt    = 0, //TC_EVT_EFFECT_NOOP,           // External event effect on TIOB.
142         .bcpc     = 0, //TC_EVT_EFFECT_NOOP,           // RC compare effect on TIOB.
143         .bcpb     = 0, //TC_EVT_EFFECT_NOOP,           // RB compare effect on TIOB.
144 
145         .aswtrg   = 0, //TC_EVT_EFFECT_NOOP, // Trigger no cambia la salida
146         .aeevt    = 0, //TC_EVT_EFFECT_NOOP, // Trigger no cambia la salida
147         .acpc     = 1, //TC_EVT_EFFECT_SET,              // RC compare effect on TIOA.
148         .acpa     = 2, //TC_EVT_EFFECT_CLEAR,          // RA compare effect on TIOA.
149 
150         .wavsel   = 2, //Simple pendiente, RC determina Periodo, RA Duty
151         .enetrg   = 0, //No hay trigger por evento externo FALSE,
152         .eevt     = 0, //No hay trigger por evento externo TC_EXT_EVENT_SEL_TIOB_INPUT,
153         .eevtedg  = 0, //No hay trigger por evento externo TC_SEL_NO_EDGE,
154         .cpcdis   = FALSE, //Se va a generar mas de un perido
155         .cpcstop  = FALSE, //Se va a generar mas de un perido
156 
157         .burst    = 0, //Sin Burst, TC_BURST_NOT_GATED
158         .clki     = 0, //Reloj no invertido, TC_CLOCK_RISING_EDGE
159         .tcclks   = 3, // fPBA/32, TC4, TC_CLOCK_SOURCE_TC4
160     };
161 
162     // Initialize the timer/counter waveform.
163     tc_init_waveform(tc, &waveform_opt);
164 }//init_tc_output
165 
```