1 //Actividad Práctica Timer 14 octubre 2019

2

3 //Bernardo Urriza A01336299 4 //Antonio Corona A01337294

5 //Fernando Cossio A00759499

6

# 7 #include <avr32/io.h> 8 #include "compiler.h"

1. #include "power\_clocks\_lib.h"
2. #include "board.h"
3. #include "gpio.h"

# 12 #include "pwm.h" 13 #include "tc.h" 14 #include "pm.h"

15 #include "intc.h"

16

1. #define BTN\_RIGHT AVR32\_PIN\_PB24
2. #define BTN\_CENTER AVR32\_PIN\_PB26

19

1. //Funciones de usuario
2. \_\_attribute\_\_ ((\_\_interrupt\_\_))

# 22 void teclas(void); 23 static void init\_tc\_input(volatile avr32\_tc\_t \*tc, unsigned int channel); 24 static void init\_tc\_output(volatile avr32\_tc\_t \*tc, unsigned int channel); 25

1. //Variables globales
2. uint8\_t state=0;
3. int ra\_input = 0, ra\_output = 0;

29

# 30 int main (void){

31

1. // Configuracion de PM
2. pm\_switch\_to\_osc0(&AVR32\_PM, 16000000, 3); //F pba= 16MHz; startup: 18ms
3. //Inicializacion de canales de captura y PWM
4. init\_tc\_input(&AVR32\_TC, 0); //Canal 0 como captura
5. init\_tc\_output(&AVR32\_TC, 2); //Canal 2 como waveform
6. //Inicializacion de GPIO por IRQ

# 38 static const gpio\_map\_t TC\_GPIO\_MAP =

1. {
2. {106, 2}, //GPIO 106, TIOA0, FN especial C, 2
3. {86, 2} //GPIO 86, FN especial C, 2
4. };
5. gpio\_enable\_module(TC\_GPIO\_MAP, sizeof(TC\_GPIO\_MAP) / sizeof(TC\_GPIO\_MAP[0]));//Activar Fn especiales para TIOA0 y TIOA2
6. gpio\_enable\_gpio\_pin(BTN\_RIGHT);
7. gpio\_enable\_gpio\_pin(BTN\_CENTER);

46

1. //Interrupciones
2. Disable\_global\_interrupt();
3. INTC\_init\_interrupts();
4. INTC\_register\_interrupt(&teclas, 71, 3); //Hab interrupcion para teclas RIGHT Y CENTER,
5. gpio\_enable\_pin\_interrupt(BTN\_RIGHT , GPIO\_FALLING\_EDGE);
6. gpio\_enable\_pin\_interrupt(BTN\_CENTER , GPIO\_FALLING\_EDGE);
7. Enable\_global\_interrupt();

54

1. //fPBA=16MHz; fPBA/32=500kHz => TPBA=2us
2. //Tpwm= 30ms => rc = Tpwm/TPBA = 15,000
3. tc\_write\_rc(&AVR32\_TC0, 2, 15000); //Periodo 30ms en pwm.
4. 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){

1. case 0: //Int tecla center
2. // deshabilitar timers
3. tc\_stop(&AVR32\_TC0,0);
4. tc\_stop(&AVR32\_TC0,2);

# 67 break;

1. case 1: //Se presiona tecla Right Comienza a capturar
2. tc\_start(&AVR32\_TC0,0); //Inicia Captura
3. tc\_start(&AVR32\_TC0,2); //Inicia PWM, trigger por SW 71 while(state == 1){ //evita que se repita el tc\_start
4. ra\_input = tc\_read\_ra(tc, 0); //Inicia con este valor el PWM
5. if (ra\_input < 4500){ //30%(15000) = 4500
6. //generar pwm con T=30ms y DC=20%
7. //raPWM = rc \* 0.2 = 3000
8. ra\_output = 3000;
9. tc\_write\_ra(&AVR32\_TC0, 2, ra\_output);
10. }else if(ra\_input <= 10500){ //70%(15000) = 10500
11. //generar pwm con T=30ms y DC=50%
12. //raPWM = rc \* 0.5 = 7500
13. ra\_output = 7500;
14. tc\_write\_ra(&AVR32\_TC0, 2, ra\_output);

# 83 }else{

1. //generar pwm con T=30ms y DC=80%
2. //raPWM = rc \* 0.8 = 12000
3. ra\_output = 12000;
4. tc\_write\_ra(&AVR32\_TC0, 2, ra\_output);
5. }
6. }

# 90 break;

1. }//Fin switch
2. }//Fin While
3. } //Fin Main

94

95 //Handler

# 96 void teclas (void) {

1. if (gpio\_get\_pin\_interrupt\_flag (BTN\_CENTER)){
2. state = 0;
3. gpio\_clear\_pin\_interrupt\_flag(BTN\_CENTER);
4. }
5. if (gpio\_get\_pin\_interrupt\_flag (BTN\_RIGHT)){
6. state = 1;
7. gpio\_clear\_pin\_interrupt\_flag(BTN\_RIGHT);
8. }
9. 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

1. // Options for capture mode.
2. tc\_capture\_opt\_t capture\_opt =
3. {
4. .channel = channel, //Canal

113

1. .ldrb = 0, //No hay carga en TC\_SEL\_NO\_EDGE,
2. .ldra = 2, //Carga en Falling de la entrada TIOA TC\_SEL\_FALLING\_EDGE, 116
3. .cpctrg = 0, //Compare con RC no detiene la captura TC\_NO\_TRIGGER\_COMPARE\_RC
4. .abetrg = 1, //Trigger por la misma TIOA TC\_EXT\_TRIG\_SEL\_TIOA 119 .etrgedg = 1, //Rising es trigger, TC\_SEL\_RISING\_EDGE 120
5. .ldbdis = FALSE, //Se va a medir mas de un periodo
6. .ldbstop = FALSE, //Se va a medir mas de un periodo

123

1. .burst = 0, //Sin Burst, TC\_BURST\_NOT\_GATED
2. .clki = 0, //Reloj no invertido, TC\_CLOCK\_RISING\_EDGE
3. .tcclks = 3, // fPBA/32, TC4, TC\_CLOCK\_SOURCE\_TC4
4. };

128

1. // Initialize the timer/counter capture.
2. tc\_init\_capture(tc, &capture\_opt);
3. }//init\_tc\_input

132

133

134 static void init\_tc\_output(volatile avr32\_tc\_t \*tc, unsigned int channel){ 135 // Options for waveform generation.

1. tc\_waveform\_opt\_t waveform\_opt =
2. {
3. .channel = channel, // Channel selection.

139

1. .bswtrg = 0, //TC\_EVT\_EFFECT\_NOOP, // Software trigger effect on TIOB.
2. .beevt = 0, //TC\_EVT\_EFFECT\_NOOP, // External event effect on TIOB.
3. .bcpc = 0, //TC\_EVT\_EFFECT\_NOOP, // RC compare effect on TIOB. 143 .bcpb = 0, //TC\_EVT\_EFFECT\_NOOP, // RB compare effect on TIOB.

144

1. .aswtrg = 0, //TC\_EVT\_EFFECT\_NOOP, // Trigger no cambia la salida
2. .aeevt = 0, //TC\_EVT\_EFFECT\_NOOP, // Trigger no cambia la salida
3. .acpc = 1, //TC\_EVT\_EFFECT\_SET, // RC compare effect on TIOA.
4. .acpa = 2, //TC\_EVT\_EFFECT\_CLEAR, // RA compare effect on TIOA.

149

1. .wavsel = 2, //Simple pendiente, RC determina Periodo, RA Duty
2. .enetrg = 0, //No hay trigger por evento externo FALSE,
3. .eevt = 0, //No hay trigger por evento externo TC\_EXT\_EVENT\_SEL\_TIOB\_INPUT,
4. .eevtedg = 0, //No hay trigger por evento externo TC\_SEL\_NO\_EDGE,
5. .cpcdis = FALSE, //Se va a generar mas de un perido 155 .cpcstop = FALSE, //Se va a generar mas de un perido

156

1. .burst = 0, //Sin Burst, TC\_BURST\_NOT\_GATED
2. .clki = 0, //Reloj no invertido, TC\_CLOCK\_RISING\_EDGE
3. .tcclks = 3, // fPBA/32, TC4, TC\_CLOCK\_SOURCE\_TC4
4. };

161

1. // Initialize the timer/counter waveform.
2. tc\_init\_waveform(tc, &waveform\_opt);
3. }//init\_tc\_output

165