Εργαστηριακή Άσκηση 03

ΠΕΡΙΓΡΑΦΗ

Η έξυπνη σκούπα θα πρέπει να ελέγχει με τον ADC για εμπόδια, έτσι θα επαναχρησιμοποιηθεί το ερώτημα 1 της προηγούμενης άσκησης. Άρχικα η συσκευή πρέπει να πάει ευθεία, να κινηθούν οι δύο πρώτες ρόδες με ίδια ταχύτητα.

Ανάλογα με το ρυθμό που αναβοσβήνουν τα Leds τέτοια ταχύτητα έχει η συσκευή. Θα χρησιμοποιηθεί ο tcb διότι έχει περισσότερες λειτουργίες. Θα χρησιμοποιήσουμε δύο tcb σε λειτουργία 8 bit PWM mode. Κάθε φορά που ο ADC θα βρίσκει τοίχο θα γίνεται interrupt και τα leds κίνησης (led0 και led1) θα απενεργοποιούνται στη ρουτίνα του adc. Μετά ανάλογα το switch που θα πατηθεί, και θα ελέγχεται στη συνάρτηση διαχείρησης interrupt (η οποία απενεργοποιεί και τους tcb) , η αντίστοιχη ρόδα θα χρησιμοποιεί τη μισή περίοδο για να κινείται πιο γρόγορα. Μόλις ξαναπατηθεί το switch θα συνεχίσει η κανονική λειτουργία.

ΚΩΔΙΚΑΣ

Εξετάζοντας τον intflags και απομονώνοντας το 5ο και 6ο μπιτ μπορούμε να καταλάβουμε πιο switch έχει πατηθεί. Ο πρώτος έλεγχος περιμένει μέχρι να πατηθεί κάποιο.

Επειδή κάθε φορά που ένας timer ενεργοποιεί το led γίνεται διακοπή και έχουμε δύο timers, θα χρησιμοποιήσουμε μια ρουτίνα iso και θα διαχειριζόμαστε και τους δύο.

Κάθε συνάρτηση που τελειώνει σε f όπως η void TCB0_initf(void), εννοεί ότι αρχικοποιεί τον timer στη fast λειτουργία με μισή δηλαδή περίοδο ρολογιού. Θα έχει αλλαγές στην τιμή των καταχωρητών στα CCMPH και CCMPL.

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define TCB_CMP_EXAMPLE_VALUE (0x80ff)

void TCB0_init (void);

void TCB0_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB0.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB0.CTRLA |= TCB_ENABLE_bm;
	TCB0.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB0.CTRLB |=TCB_CCMPEN_bm;
TCB0.CTRLB |= TCB_CNTMODE_PWM8_gc;}

void TCB1_init (void);

void TCB1_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB1.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB1.CTRLA |= TCB_ENABLE_bm;
	TCB1.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB1.CTRLB |=TCB_CCMPEN_bm;
	TCB1.CTRLB |= TCB_CNTMODE_PWM8_gc;


}

int main() {
	PORTD.DIR |= PIN0_bm;
	PORTD.DIR |= PIN1_bm;
	PORTD.DIR |= PIN2_bm;

	void TCB0_init (void);
	void TCB1_init (void);



	PORTD.OUT |= PIN0_bm;
	PORTD.OUT |= PIN1_bm;
	PORTD.OUT |= PIN2_bm;



};



ISR(TCB0_INT_vect)
{
	TCB0.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
	PORTB.IN=PIN0_bm; /*TOGGLE PB5 GPIO*/
	PORTD.OUTCLR |= PIN0_bm; //open led0
	if(TCB1.INTFLAGS==1)
	{
		TCB1.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
		PORTB.IN=PIN1_bm; /*TOGGLE PB5 GPIO*/
		PORTD.OUTCLR |= PIN1_bm; //open led1
	};
}
```

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define TCB_CMP_EXAMPLE_VALUE (0x80ff)

void TCB0_init (void);

void TCB0_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB0.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB0.CTRLA |= TCB_ENABLE_bm;
	TCB0.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB0.CTRLB |=TCB_CCMPEN_bm;
TCB0.CTRLB |= TCB_CNTMODE_PWM8_gc;}
```

```
void TCB1_init (void);

void TCB1_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB1.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB1.CTRLA |= TCB_ENABLE_bm;
	TCB1.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB1.CTRLB |=TCB_CCMPEN_bm;
	TCB1.CTRLB |= TCB_CNTMODE_PWM8_gc;


}

int main() {
	PORTD.DIR |= PIN0_bm;
	PORTD.DIR |= PIN1_bm;
	PORTD.DIR |= PIN2_bm;

	void TCB0_init (void);
	void TCB1_init (void);



	//initialize the ADC for free=running mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= ADC_FREERUN_bm; // Free-Running mode enabled
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; // The bit enable debug mode
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; //Window comparator mode
	ADC0.WINLT |= 10; //set threshold
	ADC0.INTCTRL |= ADC_WCMP_bm; // Enable Interrupts for WCM
	ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT<WINLT
	sei();//begin accepting interrupts
	ADC0.COMMAND |= ADC_STCONV_bm; // Start Conversion


	PORTD.OUT |= PIN0_bm;
	PORTD.OUT |= PIN1_bm;
	PORTD.OUT |= PIN2_bm;


};



ISR(TCB0_INT_vect)
{
	TCB0.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
	PORTB.IN=PIN0_bm; /*TOGGLE PB5 GPIO*/
	PORTD.OUTCLR |= PIN0_bm; //open led0
	if(TCB1.INTFLAGS==1)
	{
		TCB1.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
		PORTB.IN=PIN1_bm; /*TOGGLE PB5 GPIO*/
		PORTD.OUTCLR |= PIN1_bm; //open led1
	};
}
ISR(ADC0_WCOMP_vect){
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS= intflags;
	PORTD.OUT |= PIN0_bm;
	PORTD.OUT |= PIN1_bm;
	PORTD.OUTCLR |= PIN2_bm;

}







#include <avr/io.h>
 #include <util/delay.h>
 #include <avr/interrupt.h>

 #define TCB_CMP_EXAMPLE_VALUE (0x80ff)


 void TCB0_initf(void);

 void TCB0_initf(void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB0.CCMP=TCB_CMP_EXAMPLE_VALUE/2;

	/*enable tcb3 and divide clk_per by 2*/
	TCB0.CTRLA |= TCB_ENABLE_bm;
	TCB0.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB0.CTRLB |=TCB_CCMPEN_bm;
TCB0.CTRLB |= TCB_CNTMODE_PWM8_gc;}


void TCB0_init (void);

void TCB0_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB0.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB0.CTRLA |= TCB_ENABLE_bm;
	TCB0.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB0.CTRLB |=TCB_CCMPEN_bm;
TCB0.CTRLB |= TCB_CNTMODE_PWM8_gc;}

void TCB1_init (void);

void TCB1_init (void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB1.CCMP=TCB_CMP_EXAMPLE_VALUE;

	/*enable tcb3 and divide clk_per by 2*/
	TCB1.CTRLA |= TCB_ENABLE_bm;
	TCB1.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB1.CTRLB |=TCB_CCMPEN_bm;
	TCB1.CTRLB |= TCB_CNTMODE_PWM8_gc;


}
```

```
void TCB1_initF(void);

void TCB1_initF(void){
	/*load ccmp register with the period and duty cycle of the PWM*/
	TCB1.CCMP=TCB_CMP_EXAMPLE_VALUE/2;

	/*enable tcb3 and divide clk_per by 2*/
	TCB1.CTRLA |= TCB_ENABLE_bm;
	TCB1.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

	/*ENABLE PIN OUTPUT and configure configure TCB in 8-bit PWM mode*/
	TCB1.CTRLB |=TCB_CCMPEN_bm;
	TCB1.CTRLB |= TCB_CNTMODE_PWM8_gc;


}

int main() {
	PORTD.DIR |= PIN0_bm;
	PORTD.DIR |= PIN1_bm;
	PORTD.DIR |= PIN2_bm;

	void TCB0_init (void);
	void TCB1_init (void);



	//initialize the ADC for free=running mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= ADC_FREERUN_bm; // Free-Running mode enabled
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; // The bit enable debug mode
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; //Window comparator mode
	ADC0.WINLT |= 10; //set threshold
	ADC0.INTCTRL |= ADC_WCMP_bm; // Enable Interrupts for WCM
	ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT<WINLT
	sei();//begin accepting interrupts
	ADC0.COMMAND |= ADC_STCONV_bm; // Start Conversion

	PORTD.OUT |= PIN0_bm;
	PORTD.OUT |= PIN1_bm;
	PORTD.OUT |= PIN2_bm;



};



ISR(TCB0_INT_vect)
{
	TCB0.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
	PORTB.IN=PIN0_bm; /*TOGGLE PB5 GPIO*/
	PORTD.OUTCLR |= PIN0_bm; //open led0
	if(TCB1.INTFLAGS==1)
	{
		TCB1.INTFLAGS=TCB_CAPT_bm; /*CLEAR THE INTERRUPT FLAG*/
		PORTB.IN=PIN1_bm; /*TOGGLE PB5 GPIO*/
		PORTD.OUTCLR |= PIN1_bm; //open led1
	};
}



ISR(ADC0_WCOMP_vect){
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS= intflags;


	PORTD.OUT |= PIN0_bm;
	PORTD.OUT |= PIN1_bm;
	PORTD.OUTCLR |= PIN2_bm;


	PORTF.DIR |= PIN5_bm;
	PORTF.DIR |= PIN6_bm;

	while ( (PORTF.INTFLAGS  && 00100000) || (PORTF.INTFLAGS  && 00010000) ==0)




	{
		if(PORTF.INTFLAGS && 00100000 == 0010000) {  // sw6 is on
			void TCB1_initF();
		}
		else {
			void TCB0_initf();

		}
	}
}
```