***Timer concept ( Final ):***

#define *F\_CPU* 16000000UL

#include <avr/interrupt.h>

#include <avr/io.h>

#include <util/delay.h>

unsigned int i;

void pwm\_init() {

// Set PWM pin (OC1A) as output

DDRB |= (1 << DDB1);

// Fast PWM mode, non-inverted output

TCCR1A = (1 << COM1A1) | (1 << WGM10) | (1 << WGM11);

// No prescaler

TCCR1B = (1 << WGM12) | (1 << CS10);

}

int main(void) {

label:

DDRB &= ~(1 << DDB0);

PORTB |= (1 << PORTB0);

DDRC = 0xFF;

PORTC &= ~(1 << PORTC0);

i = 0; // Initial duty cycle for 50% speed

pwm\_init(); // Initialize PWM

DDRD &= ~(1 << DDD2);

PORTD |= (1 << PORTD2);

DDRD &= ~(1 << DDD3);

PORTD |= (1 << PORTD3);

// Falling edge of INT0 and INT1 generates an interrupt request

EICRA = (1 << ISC01) | (1 << ISC11);

// Enable external interrupts INT0 and INT1

EIMSK = (1 << INT0) | (1 << INT1);

sei(); // Enable global interrupts

while (1) {

OCR1A = i; // Set PWM duty cycle

*\_delay\_ms*(100); // Delay to observe the changes in speed

}

return 0;

}

// Interrupt Service Routine for INT0

ISR(INT0\_vect) {

if (i >=30)

{

PORTC &= ~(1 << PORTC0);

}

i += 30; // Increase duty cycle for higher speed

}

// Interrupt Service Routine for INT1

ISR(INT1\_vect) {

i -= 30; // Decrease duty cycle for lower speed

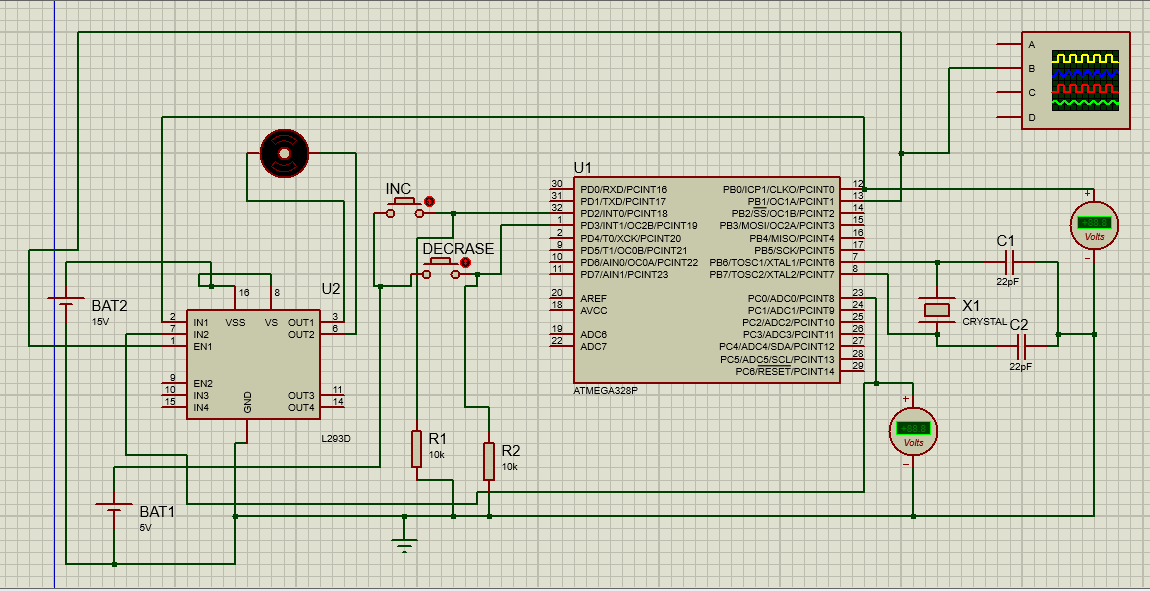
if (i < 30) {

PORTC |= (1 << PORTC0);

}

}

***Circuit :***

******

***Two Motor Controle with Buzzer :***

#define *F\_CPU* 16000000UL

#include <avr/interrupt.h>

#include <avr/io.h>

#include <util/delay.h>

int i;

void pwm\_init() {

// Set PWM pins (OC1A and OC1B) as output

DDRB |= (1 << DDB1) | (1 << DDB2);

// Fast PWM mode, non-inverted output

TCCR1A = (1 << COM1A1) | (1 << COM1B1) | (1 << WGM10) | (1 << WGM11);

// No prescaler

TCCR1B = (1 << WGM12) | (1 << CS10);

}

int main(void) {

DDRB &= ~(1 << DDB0);

PORTB |= (1 << PORTB0);

DDRC = 0xFF;

PORTC &= ~(1 << PORTC0); // out put for in2

PORTC &= ~(1 << PORTC1); // output for in4

PORTC &= ~(1 << PORTC2); // output for buzzer

i = 0; // Initial duty cycle for 50% speed

pwm\_init(); // Initialize PWM

DDRD &= ~(1 << DDD2);

PORTD |= (1 << PORTD2);

DDRD &= ~(1 << DDD3);

PORTD |= (1 << PORTD3);

// Falling edge of INT0 and INT1 generates an interrupt request

EICRA = (1 << ISC01) | (1 << ISC11);

// Enable external interrupts INT0 and INT1

EIMSK = (1 << INT0) | (1 << INT1);

sei(); // Enable global interrupts

while (1) {

OCR1A = i; // Set PWM duty cycle for OCRA

OCR1B = i \*2 ; // 130/100; // Set PWM duty cycle for OCRB to be 30% more than OCRA

*\_delay\_ms*(100); // Delay to observe the changes in speed

}

return 0;

}

// Interrupt Service Routine for INT0

ISR(INT0\_vect) {

if (i >=0)

{

PORTC &= ~(1 << PORTC0);

PORTC &= ~(1 << PORTC1);

PORTC &= ~(1 << PORTC2);

i += 30; // Increase duty cycle for higher speed

}

else if (i < 0)

{

PORTC |= (1 << PORTC0);

PORTC |= (1 << PORTC1);

PORTC |= (1 << PORTC2);

i += 30; // Increase duty cycle for higher speed

}

else

{

i += 30; // Increase duty cycle for higher speed

}

if(i==540)

{

PORTC |= (1 << PORTC2);

}

else if(i >= 1020)

{

PORTC |= (1 << PORTC2);

i=1020;

}

}

// Interrupt Service Routine for INT1

ISR(INT1\_vect) {

i -= 30; // Decrease duty cycle for lower speed

if (i < 30) {

PORTC |= (1 << PORTC0);

PORTC |= (1 << PORTC1);

PORTC |= (1 << PORTC2);

i = 0;

}

else if(i <= 1020)

{

PORTC &= ~(1 << PORTC2);

}

}

***CIRCUIT :***

