Topic 4 : TIMERS



Problem 1 : Write a C Program Using Timer 0 , Normal mode , interrupt method to toggle PORTB.4 with time interval 0.1ms using. The system use 8MHz clock, prescaler /256.

Problem 2 : Write a C Program Using Timer 0 , CTC mode , interrupt method, to toggle PORTA.2 with time interval 0.1 ms. the system use 8MHz clock, prescaler /256.

Problem 3 : Write a C Program Using Timer 1 , Normal mode , interrupt method to toggle PORTD.6 with time interval 0.5s. the system use 8MHz clock, prescaler /1024.

Problem 4 : Write a C Program Using Timer 1 , CTC mode , interrupt method, to toggle PORTC.2 with time interval 0.2s. the system use 8MHz clock, prescaler /1024.

Problem 5 : Write a C Program Using Timer 2 , Normal mode , interrupt method to toggle PORTC.3 with time interval 0.4 ms using prescaler /256. the system use 8MHz clock, prescaler /256.

Problem 6: Write a C Program Using Timer 2 , CTC mode , interrupt method, to toggle PORTB.5 with time interval 0.5 ms using prescaler /256. the system use 8MHz clock, prescaler /256.

Problem 7: Convert the C program in Problem 5 into AVR Assembly code.

**Problem 1: Toggle PORTB.4 Using Timer 0 (Normal Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer0\_init() {

// Set prescaler to /256

TCCR0 |= (1 << CS02);

// Enable timer overflow interrupt

TIMSK |= (1 << TOIE0);

// Initialize counter value

TCNT0 = 0;

// Enable global interrupts

sei();

}

ISR(TIMER0\_OVF\_vect) {

// Toggle PORTB.4

PORTB ^= (1 << PB4);

}

int main() {

// Set PORTB.4 as output

DDRB |= (1 << PB4);

// Initialize Timer 0

timer0\_init();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 2: Toggle PORTA.2 Using Timer 0 (CTC Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer0\_init\_ctc() {

// Set prescaler to /256

TCCR0 |= (1 << CS02);

// Set CTC mode

TCCR0 |= (1 << WGM01);

// Set compare value for 0.1 ms interval

OCR0 = 78; // (8 MHz / 256) \* 0.1 ms = 78

// Enable timer compare match interrupt

TIMSK |= (1 << OCIE0);

// Enable global interrupts

sei();

}

ISR(TIMER0\_COMP\_vect) {

// Toggle PORTA.2

PORTA ^= (1 << PA2);

}

int main() {

// Set PORTA.2 as output

DDRA |= (1 << PA2);

// Initialize Timer 0 in CTC mode

timer0\_init\_ctc();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 3: Toggle PORTD.6 Using Timer 1 (Normal Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer1\_init() {

// Set prescaler to /1024

TCCR1B |= (1 << CS12) | (1 << CS10);

// Enable timer overflow interrupt

TIMSK |= (1 << TOIE1);

// Initialize counter value

TCNT1 = 0;

// Enable global interrupts

sei();

}

ISR(TIMER1\_OVF\_vect) {

// Toggle PORTD.6

PORTD ^= (1 << PD6);

}

int main() {

// Set PORTD.6 as output

DDRD |= (1 << PD6);

// Initialize Timer 1

timer1\_init();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 4: Toggle PORTC.2 Using Timer 1 (CTC Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer1\_init\_ctc() {

// Set prescaler to /1024

TCCR1B |= (1 << CS12) | (1 << CS10);

// Set CTC mode

TCCR1B |= (1 << WGM12);

// Set compare value for 0.2 s interval

OCR1A = 15624; // (8 MHz / 1024) \* 0.2 s = 15624

// Enable timer compare match interrupt

TIMSK |= (1 << OCIE1A);

// Enable global interrupts

sei();

}

ISR(TIMER1\_COMPA\_vect) {

// Toggle PORTC.2

PORTC ^= (1 << PC2);

}

int main() {

// Set PORTC.2 as output

DDRC |= (1 << PC2);

// Initialize Timer 1 in CTC mode

timer1\_init\_ctc();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 5: Toggle PORTC.3 Using Timer 2 (Normal Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer2\_init() {

// Set prescaler to /256

TCCR2 |= (1 << CS22);

// Enable timer overflow interrupt

TIMSK |= (1 << TOIE2);

// Initialize counter value

TCNT2 = 0;

// Enable global interrupts

sei();

}

ISR(TIMER2\_OVF\_vect) {

// Toggle PORTC.3

PORTC ^= (1 << PC3);

}

int main() {

// Set PORTC.3 as output

DDRC |= (1 << PC3);

// Initialize Timer 2

timer2\_init();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 6: Toggle PORTB.5 Using Timer 2 (CTC Mode)**

#include <avr/io.h>

#include <avr/interrupt.h>

void timer2\_init\_ctc() {

// Set prescaler to /256

TCCR2 |= (1 << CS22);

// Set CTC mode

TCCR2 |= (1 << WGM21);

// Set compare value for 0.5 ms interval

OCR2 = 156; // (8 MHz / 256) \* 0.5 ms = 156

// Enable timer compare match interrupt

TIMSK |= (1 << OCIE2);

// Enable global interrupts

sei();

}

ISR(TIMER2\_COMP\_vect) {

// Toggle PORTB.5

PORTB ^= (1 << PB5);

}

int main() {

// Set PORTB.5 as output

DDRB |= (1 << PB5);

// Initialize Timer 2 in CTC mode

timer2\_init\_ctc();

while (1) {

// Your other tasks here

}

return 0;

}

**Problem 7: AVR Assembly Code for Problem 5**

**Assembly**

.include "m328pdef.inc"

.org 0x0000

rjmp main

.org TIMER2\_OVF\_vect

; Toggle PORTC.3

sbi PORTC, 3

reti

main:

; Set PORTC.3 as output

sbi DDRC, 3

; Initialize Timer 2

ldi r16, (1 << CS22) ; Set prescaler to /256

out TCCR2B, r16

; Enable timer overflow interrupt

ldi r16, (1 << TOIE2)

out TIMSK2, r16

; Enable global interrupts

sei

loop:

; Your other tasks here

rjmp loop