**Lab:6**

**Implement a timer.**



**MBSD Lab**

**Spring 2023**

**Submitted by:**

**Muhammad Fahad Khan [20 Pwcse1940]**

**Semester: 6**

“On my honor, as a student of University of Engineering and Technology Peshawar, I have neither nor received unauthorized assistance on this academic work”

**Submitted to:**

**Dr:Amad Khalil**

**Task1: Write a program to create 1ms exact delay through timer.**

#include <reg51.h>

// Initialize Timer 0

void initTimer0() {

// Configure Timer 0 as 16-bit mode

TMOD &= 0xF0;

TMOD |= 0x01;

// Set initial value for Timer 0

TH0 = 0xFC;

TL0 = 0x66;

// Enable Timer 0 overflow interrupt

ET0 = 1;

}

// Timer 0 interrupt handler

void timer0ISR() interrupt 1 {

// Reload the initial value for Timer 0

TH0 = 0xFC;

TL0 = 0x66;

// Your desired delay actions here

}

void main() {

// Initialize Timer 0

initTimer0();

// Enable global interrupts

EA = 1;

while (1) {

// Your main program logic here

}

}

In this example, Timer 0 is configured in 16-bit mode by setting the bits T0 and T1 of the TMOD register. The initial values of TH0 and TL0 are set to achieve a 1ms delay. These values are calculated based on the clock frequency and desired delay. Adjust them if you're using a different clock frequency or need a different delay.

The timer0ISR function is the interrupt service routine for Timer 0 overflow interrupt (interrupt vector 1). Inside this function, you can perform the desired actions that need to occur with a 1ms delay. For example, you can set flags, update variables, or perform any other necessary operations.

Remember to enable the global interrupts (EA = 1) to enable the timer interrupts. The main program loop can be used for your main program logic while the timer interrupt takes care of the 1ms delay.

**Task2: Write a program for Generation of 1 second delay using interrupt enabled Timer0**

#include <reg51.h>

// Initialize Timer 0

void initTimer0() {

// Configure Timer 0 as 16-bit mode

TMOD &= 0xF0;

TMOD |= 0x01;

// Set initial value for Timer 0

TH0 = 0xFC;

TL0 = 0x66;

// Enable Timer 0 overflow interrupt

ET0 = 1;

}

// Timer 0 interrupt handler

void timer0ISR() interrupt 1 {

static unsigned int timerCount = 0;

// Check if 1 second has passed (1000 milliseconds)

if (++timerCount == 1000) {

// Reset the count

timerCount = 0;

// Your desired actions for a 1-second delay here

}

// Reload the initial value for Timer 0

TH0 = 0xFC;

TL0 = 0x66;

}

void main() {

// Initialize Timer 0

initTimer0();

// Enable global interrupts

EA = 1;

while (1) {

// Your main program logic here

}

}

In this example, Timer 0 is configured in 16-bit mode by setting the bits T0 and T1 of the TMOD register. The initial values of TH0 and TL0 are set to achieve a 1-second delay. These values are calculated based on the clock frequency and desired delay. Adjust them if you're using a different clock frequency or need a different delay.

The timer0ISR function is the interrupt service routine for Timer 0 overflow interrupt (interrupt vector 1). Inside this function, a static variable **timerCount** is used to keep track of the number of overflows that occurred. When **timerCount** reaches 1000, indicating 1000 overflows (1 second), you can perform the desired actions for a 1-second delay.

Remember to enable the global interrupts (EA = 1) to enable the timer interrupts. The main program loop can be used for your main program logic while the timer interrupt takes care of the 1-second delay.