 **Crystal Frequency** = 11.0592 MHz.

 **Machine Cycle Frequency** = 11.0592 MHz / 12 = 921.6 kHz.

 **Machine Cycle Period** = 1 / 921.6 kHz = 1.085 μs.

**Calculation for 1-Second Delay:**

In Mode 1, Timer 1 can count up to 65536 cycles. Here's the approach to generate a 1-second delay:

1. **Max timer count**: In Mode 1, the timer counts from 0x0000 to 0xFFFF, which corresponds to 65536 counts.
2. **Timer overflow time**:
   * For 65536 counts, the overflow time is:  
     65536×1.085 μs=71.11 ms65536 \times 1.085 \, \mu s = 71.11 \, ms65536×1.085μs=71.11ms
3. **Number of overflows for 1 second**:

To get a 1-second delay, divide 1 second by 71.11 ms:

1 second/71.11 ms≈14 overflows

#include <reg51.h>

void Timer1\_Delay\_1sec(); // Function prototype

void main() {

while (1) {

Timer1\_Delay\_1sec(); // Call the delay function to generate 1-second delay

// You can toggle an LED here to observe the delay

}

}

void Timer1\_Delay\_1sec() {

unsigned int i;

// Configure Timer 1 in Mode 1 (16-bit timer mode)

TMOD |= 0x10; // Timer 1, Mode 1 (16-bit timer)

// Generate delay for 14 overflows (each overflow is approximately 71.11 ms)

for (i = 0; i < 14; i++) {

// Load Timer 1 initial value (TH1 = 0x00, TL1 = 0x00)

TH1 = 0x00;

TL1 = 0x00;

// Start Timer 1

TR1 = 1;

// Wait for Timer 1 to overflow (TF1 == 1)

while (TF1 == 0);

// Clear overflow flag for the next overflow

TF1 = 0;

// Stop Timer 1

TR1 = 0;

}

}

Mode 2:

**Calculation for 1-Second Delay:**

To achieve a 1-second delay, we will break the time into smaller chunks using Timer 1's 8-bit auto-reload capability:

1. **Timer overflow time**: Timer 1 is an 8-bit timer, so it can count from 0x00 to 0xFF (255 in decimal). To reduce the count, we can preload the timer with an initial value.
2. **Timer 1 in Mode 2 will count from the loaded value to 0xFF**. Let’s preload it with 0x00 for maximum counting (256 counts):
   * Number of machine cycles for Timer 1 to overflow = 256 cycles.
   * Timer overflow time = 256 × 1.085 μs = 277.76 μs.
3. **Total number of overflows for 1 second**:
   * To achieve a 1-second delay, divide 1 second by the overflow time:
   * 1 second/277.76 μs≈3601 overflows

#include <reg51.h>

void Timer1\_Delay\_1sec(); // Function prototype

void main() {

while (1) {

Timer1\_Delay\_1sec(); // Call the delay function to generate 1-second delay

// You can toggle an LED here to observe the delay

}

}

void Timer1\_Delay\_1sec() {

unsigned int i;

// Configure Timer 1 in Mode 2 (8-bit auto-reload)

TMOD |= 0x20; // Timer 1, Mode 2 (8-bit auto-reload)

TH1 = 0x00; // Load the timer register with 0x00 for maximum count (256 counts)

// Start Timer 1

TR1 = 1;

// Generate delay for 3601 overflows

for (i = 0; i < 3601; i++) {

while (TF1 == 0); // Wait for Timer 1 overflow (TF1 = 1)

TF1 = 0; // Clear overflow flag for the next overflow

}

// Stop Timer 1

TR1 = 0;

}