Chapter 4

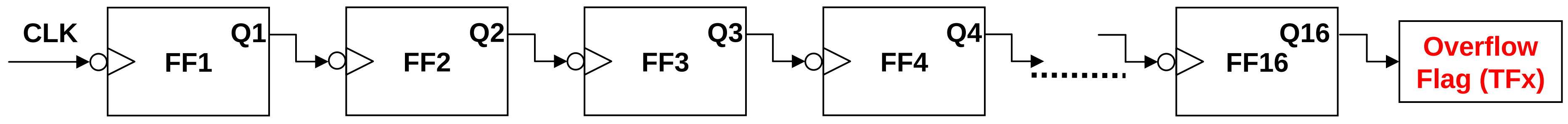
Timer Operation

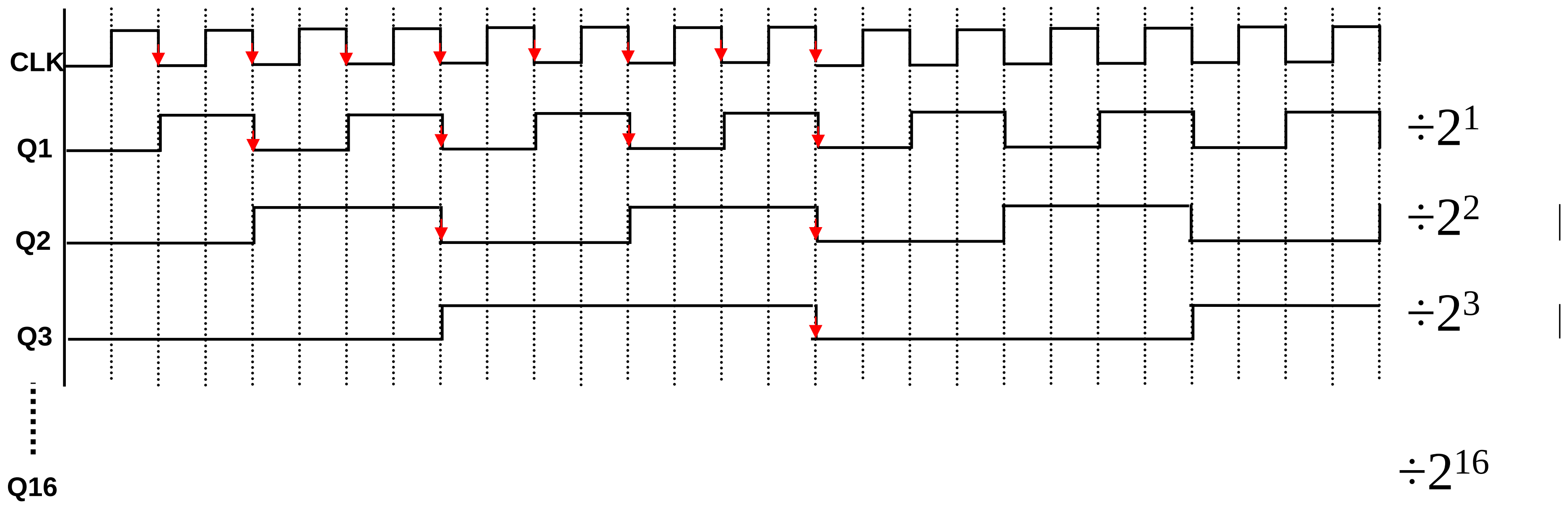
(I. Scott MacKenzie)

Counter / Timers (T0 & T1)

- Timer is a series of divide-by-two flip-flops that receive an input signal as a clocking source.
- Clock is applied to the first flip-flop, which gives output divided by 2.
- That output of first flip-flop clocks the second flip-flop,which also divides it by 2 and so on.
- The output of the last stage clocks a timer overflow flip-flop, or flag, which is tested by the software.
- It is like a counter. A 16-bit timer would count from 0000H to FFFFH. The overflow flag is set on the FFFFH-to-0000H count.
- There are two timers in 8051 i.e., T0 and T1.
- Four modes of timer operations.

Divide-by-Two Flip-Flops





TMOD : Counter/Timer MODe Register

GATE	**$C/\overline{T}$**	**M1**	**M0**	**GATE**	**$C/\overline{T}$**	**M1**	**M0**
Timer 1				**Timer 0**			

• TMOD is not bit addressable. It is loaded, generally, by the software at the beginning of a program to initialize the timer mode.

- GATE : Permits INTx pin to enable/disable the counter.

- $C/\overline{T}$: Set for counter operation, reset for timer operation.

- M1, M0 : To Select the Mode

    00 : Mode 0 - 13-bit timer mode (Emulates 8048).

    01 : Mode 1 - 16-bit timer mode.

    10 : Mode 2 - 8-bit auto-reload mode.

    11 : Mode 3 - Split timer mode (Timer 0 = two 8-bit timers).

TCON : Counter/Timer CONtrol Register

$$\underbrace{\text{TF1}\quad\text{TR1}\quad\text{TF0}\quad\text{TR0}}_{\text{Timers}}\quad\underbrace{\text{IE1}\quad\text{IT1}\quad\text{IE0}\quad\text{IT0}}_{\text{Interrupts}}$$

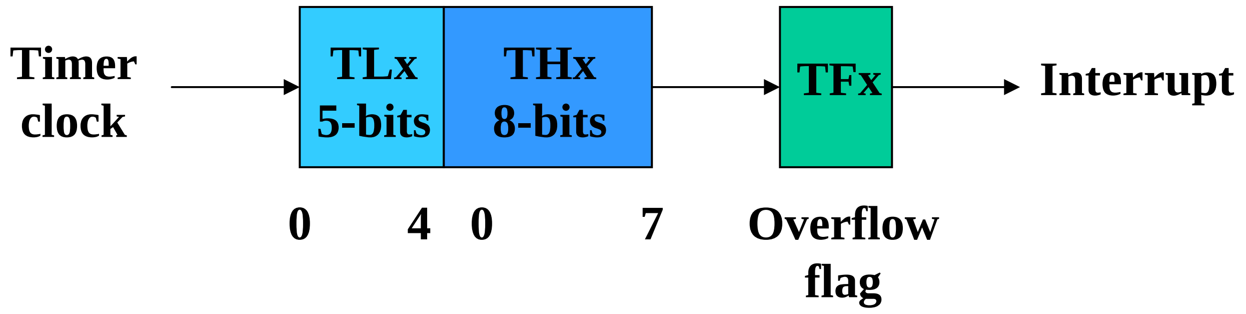
- TF1, TF0 : Overflow flags for Timer 1 and Timer 0.
- TR1, TR0 : Run control bits for Timer 1 and Timer 0.
  Set to run, reset to hold.

- IE1, IE0 : Edge flag for external interrupts 1 and 0. *
  Set by interrupt edge, cleared when interrupt is processed.
- IT1, IT0 : Type bit for external interrupts. *
  Set for falling edge interrupts, reset for 0 level interrupts.

* = not related to counter/timer operation but used to detect and initiate external interrupts.

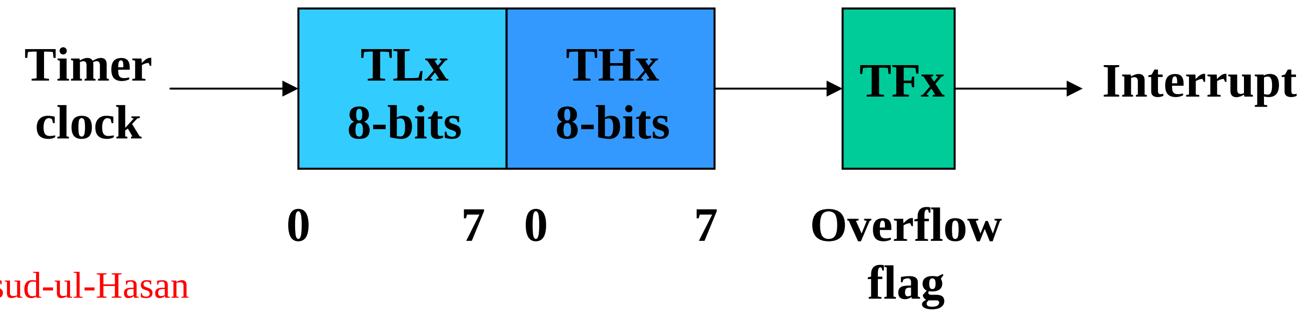
Timer Modes

- **Timer Mode 0 (13-bit Timer):**
  - Timer high-byte (THx) is cascaded with the 5 least-significant bits of the timer low-byte (TLx) to form a 13-bit timer, where x = 0 or 1.
  - Upper 3-bits of TLx are not used.
  - Overflow occurs on the 1FFFH-to-0000H and sets the timer overflow flag.
  - MSB is THx bit 7, and LSB is TLx bit 0.
  - MOV TMOD, #00H ; setting both timers to mode 0



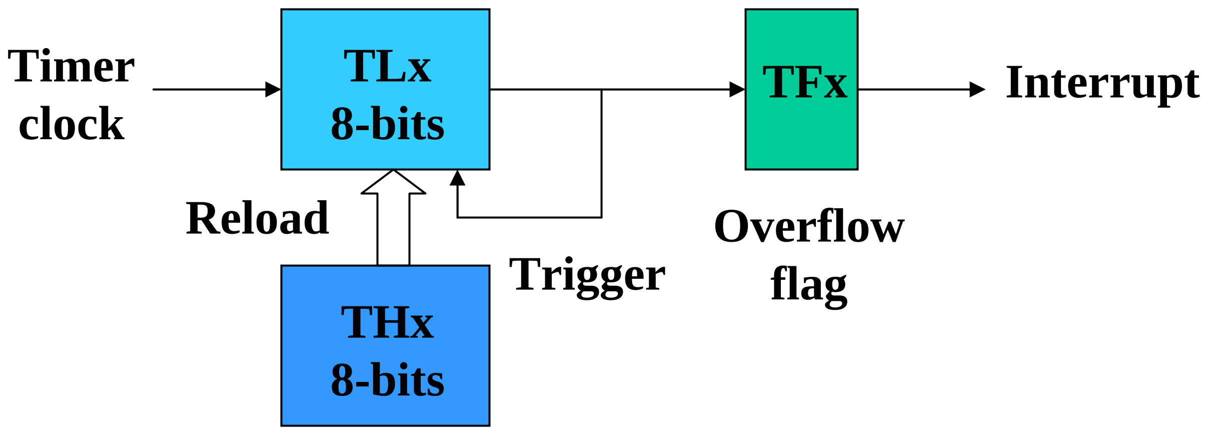
Timer Modes (cont'd)

- **Timer Mode 1 (16-bit Timer):**
  - Same as mode 0 except that it is 16-bit. Timer high-byte (THx) is cascaded the timer low-byte (TLx) to form a 16-bit timer, where x = 0 or 1.
  - Clock is applied to the combined high and low-byte timer registers.
  - Overflow occurs on the FFFFH-to-0000H and sets the timer overflow flag.
  - MSB is THx bit 7, and LSB is TLx bit 0.
  - LSB toggles at ***clock frequency/2^1*** and MSB at ***clock frequency/2^{16}***



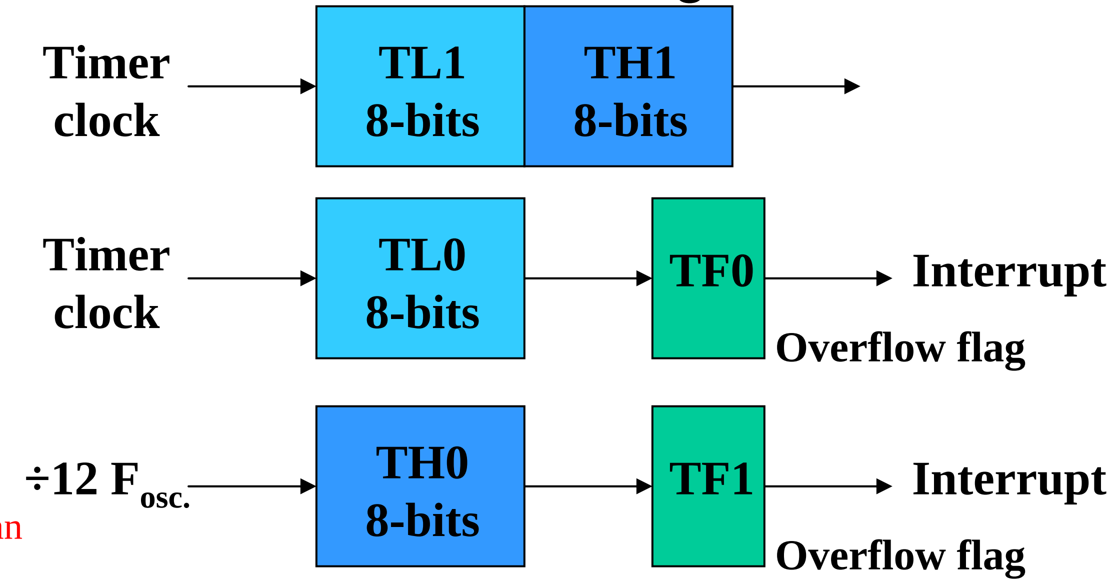
Timer Modes (cont'd)

- **Timer Mode 2 (Auto-Reload):**
  - Timer low -byte (TLx) operates as an 8-bit timer while the timer high -byte (THx) holds a reload value.
  - When the count overflows from FFH-to-00H, not only the timer flag set, but also the value in THx is loaded into TLx, and counting continues from this value up to next FFH-to-00H, so on.



Timer Modes (cont'd)

- **Timer Mode 3:**
  - Timer 0 Splits into two 8-bit counter/timers. TL0 and TH0 act as two separate timers with overflows setting the TF0 and TF1 respectively.
  - Timer 1 (when timer 0 is in mode 3):
    - Counter stopped if in mode 3
    - Can be used in mode 0, 1, or 2
    - Has gate (INT1) and external input (T1), but no flag or interrupt.
    - May be used as a baud rate generator.



Clocking Sources

1. Interval Timing
2. Event Counting

Clocking Sources (contd.)

1. Interval Timing:
   - If $C/\overline{T} = 0$ (in TMOD), timer operation is selected and timer is clocked from on-chip oscillator. A divide-by-12 clock frequency is applied.
   - Timer registers (TLx/THx) increment at a rate of 1/12th the frequency of on-chip oscillator.
   - 12 MHz crystal would yield a clock rate of 1 MHz.
   - Timer overflows occur after a fixed number of clocks, depending on the initial value loaded into the timer registers.

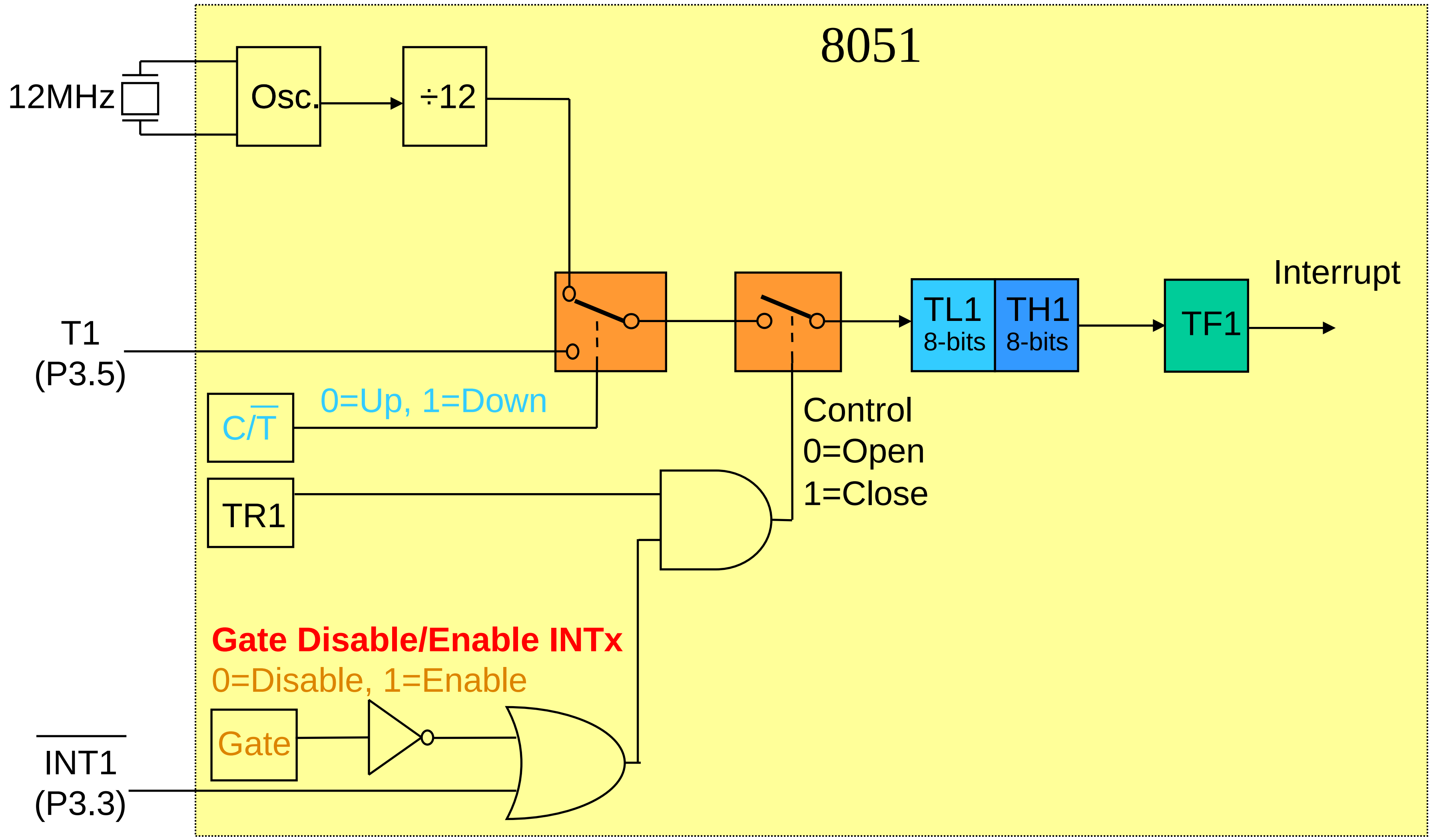
Clocking Sources (contd.)

2. Event Counting:
   - If $C/\overline{T} = 1$ (in TMOD), counter operation is selected and timer is clocked from external source. Usually, external source supplies the timer with a pulse upon the occurrence of an event. Timer counts those events.
   - External clock source comes through P3.4 (for Timer 0) and P3.5 (for Timer 1).
   - Timer registers are incremented in response to a 1-to-0 transition at the external input.
   - Number of external events is determined in software by reading the timer registers TLx/THx.

Start, Stop, and Control of Timers

- TRx bit in bit addressable register TCON is responsible for starting and stopping the counters
  - TRx = 0 stops/disables the timers (e.g., CLR TR1)
  - TRx = 1 starts/enables the timers (e.g., SETB TR0)
- System reset clears TRx, so timers are disabled by default.

Timer 1 Operating in 16-bit (Mode 1)



Initializing and Accessing Timer Registers

- Timers are usually initialized once at the beginning of the program to set the correct operating mode.
- Then, within the body of a program, the timers are started, stopped, flag bits are tested and cleared, timer registers read or updated, and so on, as required in the application.
- First register to be initialized is TMOD to set the mode of operation e.g.,

  MOV TMOD, #00010000B ; sets Timer 1 in mode 1, leave $C/\overline{T}$ = 0 and GATE = 0 for internal clocking, and clears the Timer 0 bits.

Initializing and Accessing Timer Registers (Contd.)

- Secondly, registers to be initialized are TLx/THx. E.g., For 100μs interval, the following instruction will do the job

  MOV TL1, #9CH ; $(-100)_{10}$ = FF9CH
  MOV TH1, #FFH ; load Timer 1 registers by FF9CH

Initializing and Accessing Timer Registers (Contd.)

- The timer is then started by setting the run control bit i.e.,

  ```
  SETB    TR1
  ```

- Overflow flag is automatically set 100µs later. Following instruction will check that

  ```
  WAIT:   JNB TF1, WAIT ; wait until overflow flag is set.
  ```

- When the timer overflows, it is necessary to stop the timer and clear the overflow flag in software by the following instructions:

  ```
  CLR TR1; stop Timer 1
  CLR TF1; clear overflow flag of Timer 1
  ```

Short and Long Intervals

- Shortest possible interval is one machine cycle i.e., 1µs (using 12 MHz crystal frequency).
- An 8-bit counter can give maximum delay of 256µs because $2^8=256$.
- A 13-bit counter can give maximum delay of 8192µs because $2^{13}=8192$.
- A 16-bit counter can give maximum delay of 65536µs because $2^{16}=65536$.

  65536µs = 0.065536 sec = 0.066 sec (approx.)

Short and Long Intervals (Contd.)

- For more than 0.066 sec delay, there are two methods:

1. Cascade Timer 0 and Timer 1 but in this way both timers will be tied up.
2. Use one timer in 16-bit mode with a software loop counting overflows.

Ex4-1: Pulse Wave Generation

Write a program that creates a periodic waveform on P1.0 with as high a frequency as possible. What are the frequency and duty cycle of the waveforms?

```
        ORG   0000H
LOOP:   SETB  P1.0      ; one m/c cycle
        CLR   P1.0      ; one m/c cycle
        SJMP  LOOP      ; two m/c cycle
        END
```

Frequency = 1/4µs = 250kHz

Duty Cycle = 25%

Ex4-2: 10 kHz Square Wave

Write a program using Timer 0 to create a 10 kHz square wave on P1.0.

```
        ORG   0000H
        MOV TMOD, #02H ;auto-reload mode
        MOV TH0, # -50      ;T= 1/10kHz=100µs
        SETB  TR0           ;start timer
LOOP:   JNB  TF0, LOOP      ;wait for overflow
        CLR  TF0            ;clear overflow flag
        CPL   P1.0          ;toggle port bit
        SJMP LOOP           ;repeat
        END
```

Ex4-3: 1 kHz Square Wave

Write a program using Timer 0 to create a 1 kHz square wave on P1.0.

```
        ORG   0000H
        MOV TMOD, #01H ;16-bit mode
LOOP:   MOV TH0, # FEH   ;T= 1/1kHz=1000µs
        MOV TL0, # 0CH   ;-500=FE0CH
        SETB  TR0        ;start timer
WAIT:   JNB  TF0, WAIT   ;wait for overflow
        CLR  TR0         ;stop timer
        CLR  TF0         ;clear overflow flag
        CPL   P1.0       ;toggle port bit
        SJMP LOOP        ;repeat
        END
```

Ex4-4: Buzzer Interface

Write a program that sounds the buzzer connected at P1.7 for 1 sec for each 1-to-0 transition detected at P1.6.

```
            ORG   0000H
            MOV TMOD, #01H ;16-bit mode
LOOP:       JNB  P1.6, LOOP      ;wait for 1 input
LOOP1:      JB  P1.6, LOOP1      ;wait for 0 input
            SETB  P1.7           ;turn buzzer ON
            CALL   DELAY         ;wait for 1 sec
            CLR  P1.7            ;turn buzzer OFF
            SJMP LOOP            ;repeat
```

Ex4-4: Buzzer Interface (contd.)

```
DELAY:      MOV R7, #100        ;for 100 loops
AGAIN:      MOV TH0, #D8H       ;10000µsX100=1sec
            MOV TL0, #F0H       ;-10000=D8F0H
            SETB TR0
WAIT:       JNB  TF0, WAIT
            CLR  TF0
            CLR  TR0
            DJNZ  R7, AGAIN
            RET
            END
```