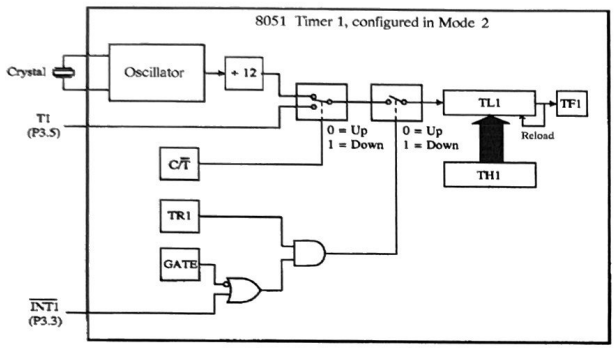
How can we make some precisely timed square waves using R31JP?

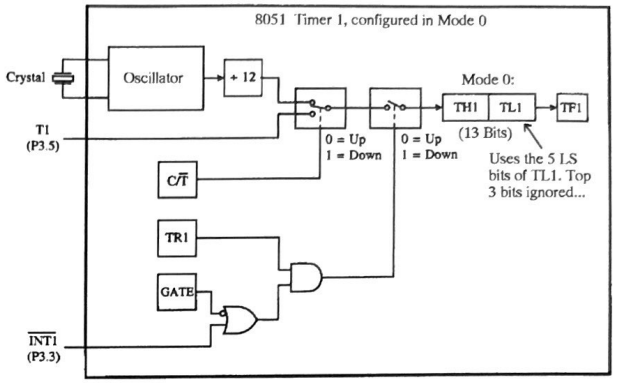
Class happy quiz: think about some short programs that make fast square waves. Today: How might we use timers to help? Our familiar tool, a timer in auto-reload mode. Very helpful for our serial port work.

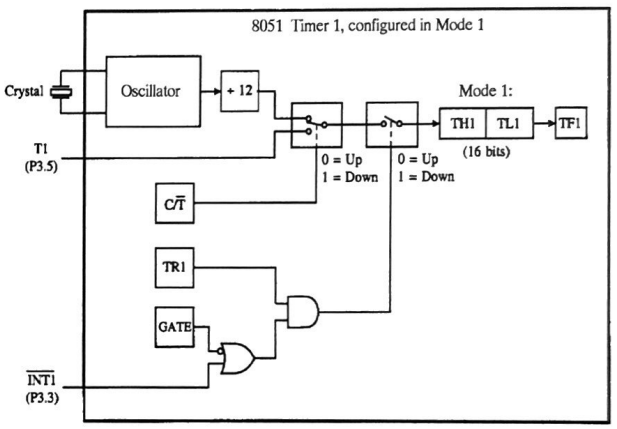


Here's a program for a 10kHz square wave on the P1.0: (assume a 12MHz crystal)

```
MOV TMOD, #02H
MOV TH0, #0Ceh  ; 50 counts
SETB TR0
LOOP:
   JNB TF0, LOOP
   CLR TF0
   CPL P1.0
SJMP LOOP
```

Will this give a frequency of exactly 10kHz?





TMOD Register

Timer/Counter 1				Timer/Counter 0			
GATE#	C/T#	M1	M0	GATE#	C/T#	M1	M0

M0 and M1 (bits 0 and 1 of TMOD, respectively): *Mode Select*
The 2-bit field (M1,M0) selects one of four modes. Mode 0 is a 13-bit counter. An interrupt is generated when the counter overflows. Thus it takes 2^{13} or 8192 input pulses to generate the next interrupt. Mode 1, similar to Mode 0, implements a 16-bit counter. It takes 2^{16} or 65,526 input pulses to generate the next interrupt. Mode 2 operates in an 8-bit reload fashion. TLi serves as an 8-bit timer/counter. When the counter overflows, the number stored in THi is copied into TLi and the count continues. An interrupt is generated each time the counter overflows and a reload is performed. In Mode 3, Timer 1 is inactive and simply holds its count. Timer 0 operates as two separate 8-bit timers. TL0 is controlled by Timer 0 control bits and generates a Timer 0 interrupt at overflow. TH0 operates as a timer driven by the system clock, prescaled by 12, and causes a Timer 1 interrupt at overflow.

C/T# (bit 2): *Counter/Timer Select*
When set, the timer/counter operates as a counter. The count increment is caused by external pulses through the T0 pin—the alternative function of P3.4. When cleared the timer/counter operates as a timer. The count increment is caused by every 12th system clock pulse. That is, the input to the counter is the system clock prescaled by 12.

GATE# (bit 3): *Gate*
Provided that TR0 of TCON (see next page) is set, clearing GATE enables (starts) counter/timer operation. If GATE is set, again provided that TR0 is set, the timer/counter operation is enabled if INT0# is high.

TCON Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0 (bit 0): *Interrupt 0 Type*
External interrupt 0 is received through bit 2 of Port 3 as an alternative function assignment. Setting IT0 causes the INT0# to be recognized at the falling edge of the input signal. Clearing IT0 causes an interrupt to be generated when the external signal is low. If the external signal stays low, INT0 will be generated over and over if IT0 is cleared but will not be generated if IT0 is set. IT0 is completely under software control.

IE0 (bit 1): *Interrupt 0 Edge Flag*
Set by hardware when an external interrupt edge is detected. Cleared when a Return From Interrupt (RETI) instruction is executed.

IT1 (bit 2): *Interrupt 1 Type*
IT1 is associated with External Interrupt 1. Its function is similar to that of IT0 as described above.

IE1 (bit 3): *Interrupt 1 Edge Flag*
IE1 is associated with External Interrupt 1. Its function is similar to that of IE0 as described above.

TR0 (bit 4): *Timer 0 Run Control Bit*
Timer/Counter 0 is disabled when TR0 is cleared. Setting TR0 is necessary but not sufficient to enable Timer/Counter 0 (see GATE# and INT0#). TR0 is completely under software control.

TF0 (bit 5): *Timer 0 Overflow Flag*
TF0 is set by hardware when Timer/Counter 0 overflows. TF0 is cleared by hardware when the processor branches to the associated interrupt service routine. TF0, along with IE0, may be used by software to determine the state of the timer.

TR1 (bit 6): *Timer 1 Run Control Bit*
TR1 is associated with Timer/Counter 1. Its function is similar to that of TR0 described above.

TF1 (bit 7): *Timer 0 Overflow Flag*
TF1 is associated with Timer/Counter 1. Its function is similar to that of TF0 described above.

Techniques for programming timed intervals:
(assume 12Mhz clock)

TECHNIQUE	INTERVAL in microsecs
Software tuning (no timers)	~10
8 bit auto-reload timer	256
16 bit timer	65,536
8 or 16 bit timer with software loops	Very long

How about a 1kHz square wave on P1.0? 500usec half period, longer than the 256 usec available with mode 2.
Try mode 1 (16 bit timer): (No autoreload, so timing is imperfect…)

```
MOV TMOD #01h
LOOP:
 MOV TH0, #0Feh ;-500 high byte
 MOV TL0, #0Ch  ;-500 low byte
 SETB TR0
 WAIT:
    JNB TF0, WAIT
 CLR TR0
 CLR TF0
 CPL P1.0
 SJMP LOOP
```

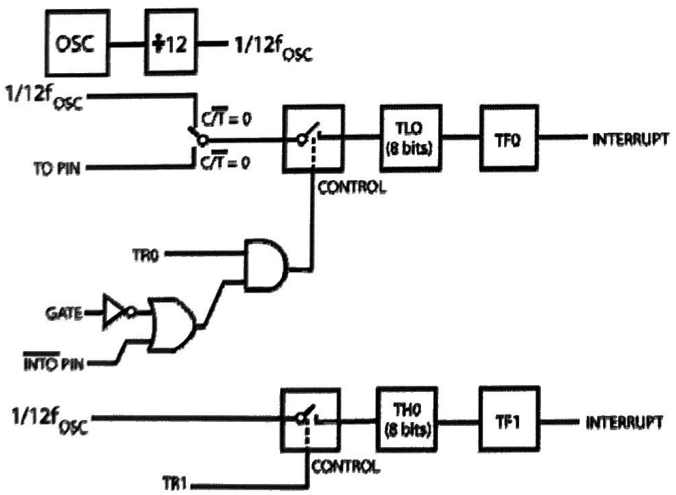

How about a 2 second period? (12 MHz crystal)

```
mov TMOD,#01h
 loop:
      cpl P1.0
      lcall delay
      sjmp loop

 delay:
      mov R7,#100
      again:
         mov TH0, #0D8h
         mov TL0, #0F0h
         setb TR0
         wait2: jnb TF0, wait2
         clr TF0
         clr TR0
         djnz R7, again
      ret
```

D8h and F0h are the high and low bytes for -10000

And, finally, here's MODE 3 (for timer 0):



A brief introduction to interrupts - another way to a 10kHz square wave:

```
.org 0
ljmp MAIN
org 000Bh
T0ISR:
    cpl P1.0
    reti
.ORG 0030h
MAIN:
    mov TMOD,#02h
    mov TH0, #0Ceh ;50 counts
    setb TR0
    mov IE,#82h
    loop: sjmp loop
```