ES – there are numerous apps where we need to generate precise timing or count events like

# of persons entering the room

# of persons watching cricket match

Situations where we have to generate precise clock or time delays

All these can be efficiently done by timers and counters

approaches

Sw based approach – processor is used for generating precise clock, time delays , count events and remains busy for the above purposes

Very flexible but processor remains busy and cannot do anyother useful work

So cpu is tied for one job only

Purely hw based approach

Extra hw used to generate clock, delay and count events

Not very flexible

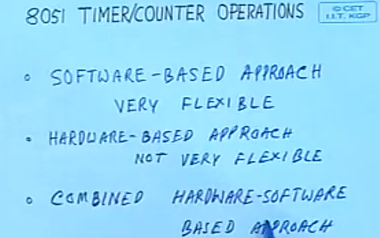
There are apps where cpu has to do other jobs along with clock, timer delay and count events

Best alternative is combined hw sw approach

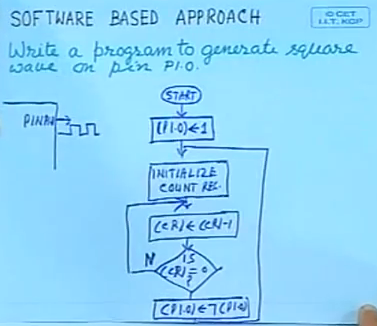
Use extra hw and some sw for clock, delay and counting

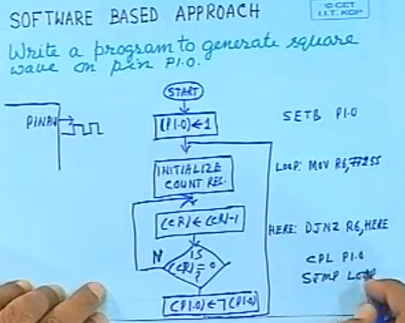
Flexible like sw approach and cpu is not tied down only for this job

Implemented in an efficient way



SW approach





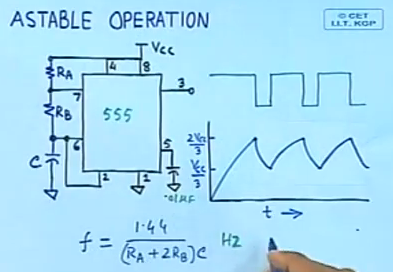
By this above code we can generate a clock and the freq is decided by the value loaded into the register

But cpu is busy just for doing the delay operation

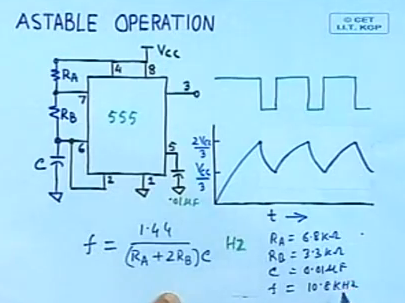
Sw approach is flexible for generating any delay by changing the freq value of the register but cpu is busy

We can use this when uc is used just for delay purpose

For multi activities we need to use sw and extra hw approach like 555 timer



We can also use a 555 timer where the Registers and capacitors decides the clock frequency

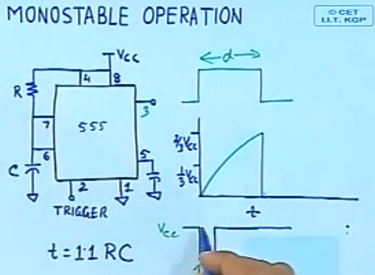


For different frequencies we need to the change the R and C values

In this hw approach we need an extra hw and simple but not at all flexible since we need to change R and C values

Also R and C values are not stable – they may change with temperature, time etc

So we cannot generate precise clock using precise components like R and C



We can also generate delay using monostable operation by a trigger in 555 timer

Here also we choose values of R and C to get different delays

Here also accuracy of the delay depends on the values of R and C which are not very accurate and stable

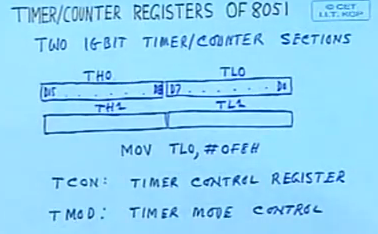
**Combined SW and HW approach**

8051 designers have provided built in approach for this

2 16b timers counters sections are built in

Provided as 4 8b registers

* Atleast 2 Timers – T0 and T1 (8052 – additional Timer T2)
* 16b Timers – hold (0 -65535) values
* Incremented periodically: 8051 - every 12 osc cycles
* 8051 OSC – 12MHz => 1 million increments per second
* Used to
  + measure intervals of time
    - For example, we can measure the duration of a function by comparing the duration at begin and end
  + generate precise hardware delays
  + generate ‘time out’ facilities - a key requirement in systems with real-time constraints



TCON – dual purpose register

High order 4 bits are for timer control

Low order 4 bits are for interrupt control

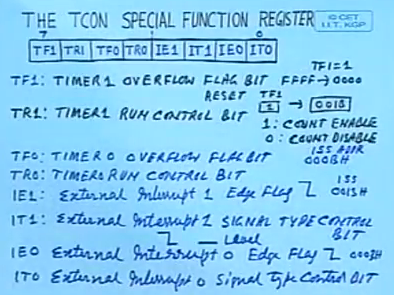
When the timer registers are initialized and started, they count in upward

In context of 8253 the counter operates in down fashion and terminal count is reached when all are 0

In 8051, it is up counter, so after it reaches ffff – 0000 – means overflow

TF1 flag can be reset either by sw using rst instn or c program or whenever we jump to the ISS of timer1 at 001BH then it will auto gets reset

Counting operation can be controlled by sw by TR bits (enable or disable)



TMOD SFR

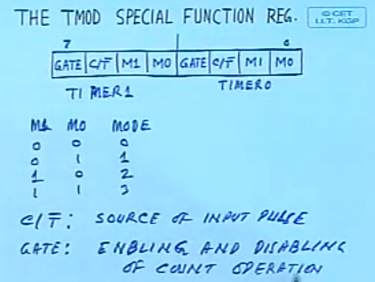
c/t – is to define the source of input pulse

c = 1 – source comes from external pulse

t=0 => source is internal

gate – define enabling and disabling of count operation

whether it is from sw means or hw means



Clock (OSC) is divided by 12 and no of clock cycles per machine cycle is also 12

So essentially in timer mode the counter is counting the no of machine cycles

c/t = 0 means timer mode and clock comes internally

OSC -> 12MHz divide by 12 -> 1MHz input to the counter

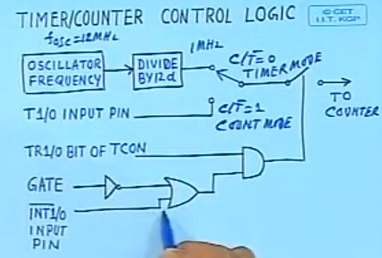
c/t=1 means counter mode and clock pulse comes from the Timer 1 and 0 input pins

basically we are counting the external events

now for disabling or enabling of count by software we use a circuit as shown and gate input of TMOD register

gate bit = 0 sw control of timer mode operation

gate bit = 1 – hw control of timer from interrupts 0/1 pints - to control timer mode operation

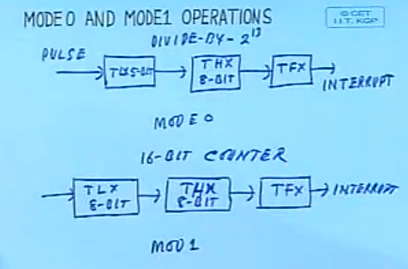


So 8051 provides both sw and hw means of controlling the timer and counter operations

**Operation modes of timer counter section**

Mode 0 – divide by 2 power 13 or 13b divider

Or prescaling by 32 and divide by 8b



Mode 0 – 13b timer or counter

Mode 1 – 16b timer or counter

**Mode 2 operation**

**Counters can be used as a 8b counter**

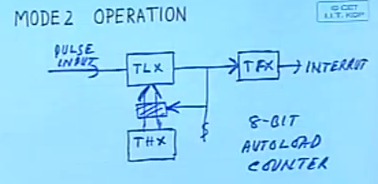
In prev mode 0 and mode1 modes once overflow occurred to start fresh we have to re init the timers

But mode 2 is auto reload – init done automatically

We have to init the THX which is reloaded into TLX each time overflow occurs from TLX

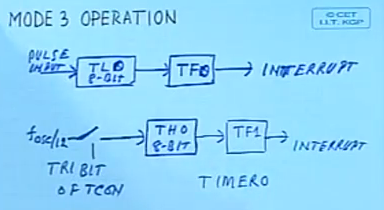
This is usefull for repetitive apps

This can be used for baud rate generation for serial data tx



**Mode 3 operation**

We can have 2 separate 8b counter from timer 0



In mode 3 we can operate timer 0 as two 8b counters

And since we use timer1 TR1 and TF1 – timer 1 will not be used or operate in mode 3

Timer 1 cannot be used in its usual operation

So we use timer 0 in mode 3 only when we require 2 separate 8b counters and timer 1 does not require TR1 and TF1 flag bits

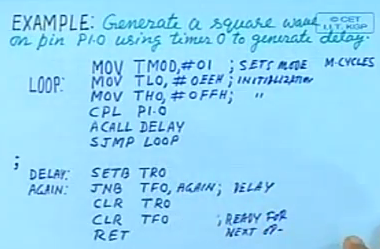
In such case timer 1 can be used to generate baud rate in mode 2 auto load mode

In this mode 2 for timer 1 TR1 and TF1 are not necessary

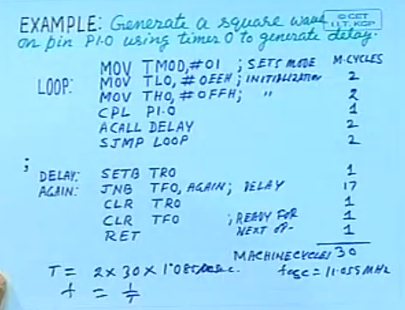
So we can use them for timer 0 and we get 2 separate 8 b counters

Essentially we get 3 8b counters – 2 from timer 0 in mode 3 and 1 from timer 1 in mode 2

**Application of timer counter section**



The above code generates a square wave and lets see the freq of the clock

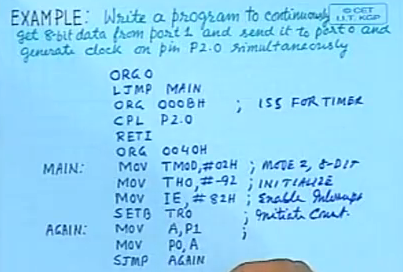


We have used timer0 in poll mode – sw approach – async approach – to check the TF 0 flag for Of

We can also use it in interrupt driven mode

In interrupt mode we can continue operations as well as generate the clock delay

Clock generated on a pin and at same time some other operations are also done



Here we haveused interrupt mode of timer 0 – so that we generate clock on a pin as well as the cpu reads and displays the data on different ports

So cpu is not tied down only generating the delay – checking status of TF0

Above all are timer operation in timer mode

We can also use it in counter mode – event counting

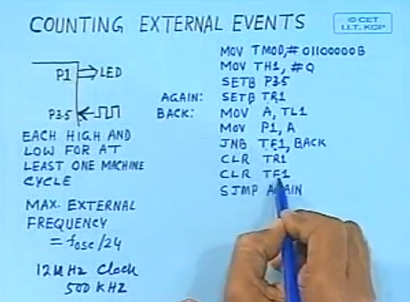
Counting external events using timer counter

Count the # of events taking place on a port pin

Sensing the clocks at the port pin

What is the max rate of sensing?

Decided by the f0sc clock



In this mode counteris used for counting events

Ex: conference room, cricket match spectators

8051 has built in 2 16b timer/counter section with 4 modes of operation