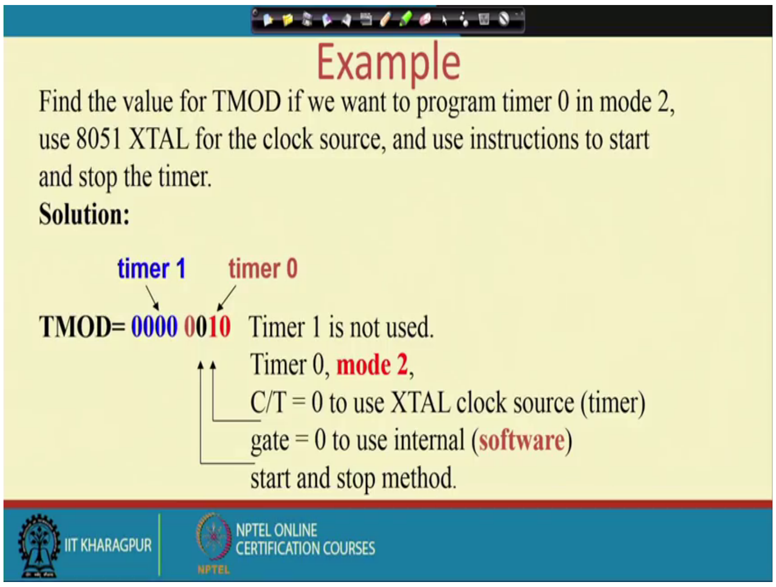
Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 32
8051 Microcontroller (Contd.)

Next we look into an example, suppose we want to find the value for setting of this TMOD register, if you want to program this timer 0 in mode 2 and use the 8051 crystal frequency as the clock source, and use the instructions to start and stop the timer.

(Refer Slide Time: 00:19)



Example

Find the value for TMOD if we want to program timer 0 in mode 2, use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

Solution:

timer 1
timer 0

TMOD= 0000 0010 Timer 1 is not used.

Timer 0, **mode 2**,

C/T = 0 to use XTAL clock source (timer)

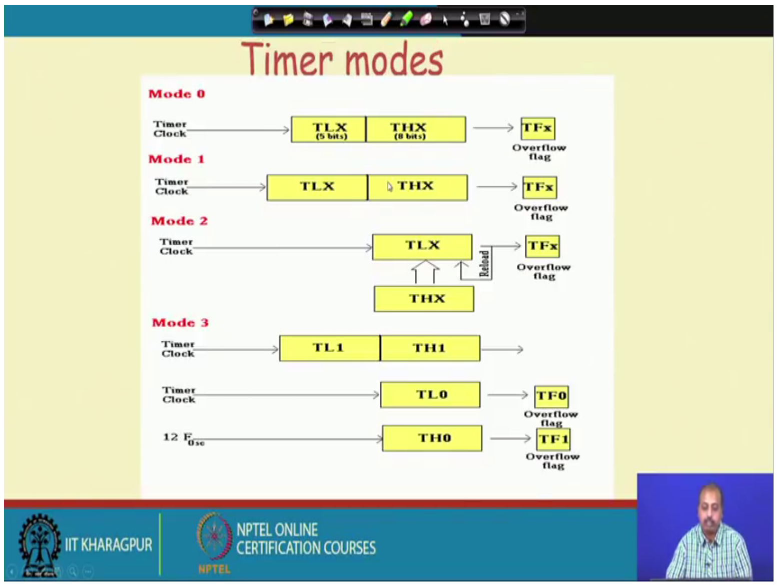
gate = 0 to use internal (**software**) start and stop method.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, basically it is a soft software-controlled timer that we want to operate, and we want to use this timer 0 in mode 2. So, definitely this TMOD setting. So, time for timer 1 this 4 most significant bits are for timer 1. So, they are all 0, now for this least significant 4 bit so, they are for timer 0, and out of that this gate bit is 0. So, this gate equal to 0, that is a internal time internal operation, then this is your this count..

So, that is for then this second bit is C by T. So, C by T is 0, means we are looking for the timer operation. So, C by T is 0 and then 1 0. So, this is the mode identifier. So, this is the pattern that we have to load in the TMOD register, if we want to do this operation.

(Refer Slide Time: 01:26)

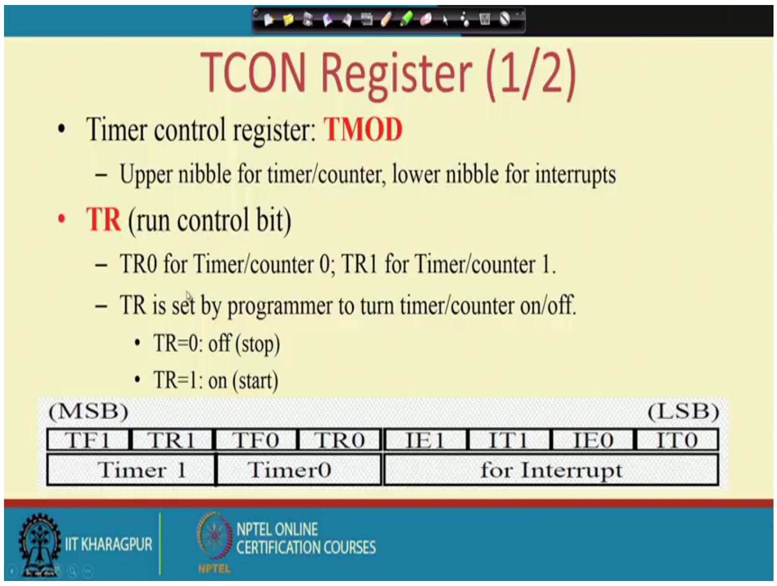


So, this is the summary of this timer modes. So, in mode 0 what happens is that. So, this timer clock. So, this this comes from the internal source, and then this TLX has got 5 bits, and THX has got 8 bits, and when it overflows then this TFx bit will be set. So, there is a for timer 0. So, there is a TF 0 bit, which is basically the overflow bit. So, when this timer overflows this bit is set.

In mode 1. So, that was a 16-bit mode. So, again the same thing TLX and THX. So, this is in mode 0 it was 13 bit here it is a 16-bit timer, and again the rest of the thing is same. So, it is it will when it overflows. So, it will affect the TF0 bit, then in mode 2 so, we have got this reload mechanism. So, this when this TLX is. So, it is an 8-bit timer. So, when the timer overflows the TFx is set, and after that the value is reloaded from THX into TLX and the process restarts. So, this will be doing a periodically, in mode 3 we have got this 2 such timers TL 0 and TH 0. So, there are 2 overflow flags TF 0 and TF 1..

Now, you see that this timer clock. So, when TL 0 will overflow. So, this TF 0 will be set, and this when this TH 0 workflows then this is TF 1 will be set. So, we will come to more detail with mode 3 later. So, we will see that later.

(Refer Slide Time: 03:11)



TCON Register (1/2)

- Timer control register: **TMOD**
  - Upper nibble for timer/counter, lower nibble for interrupts
- **TR** (run control bit)
  - TR0 for Timer/counter 0; TR1 for Timer/counter 1.
  - TR is set by programmer to turn timer/counter on/off.
    - TR=0: off (stop)
    - TR=1: on (start)

(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

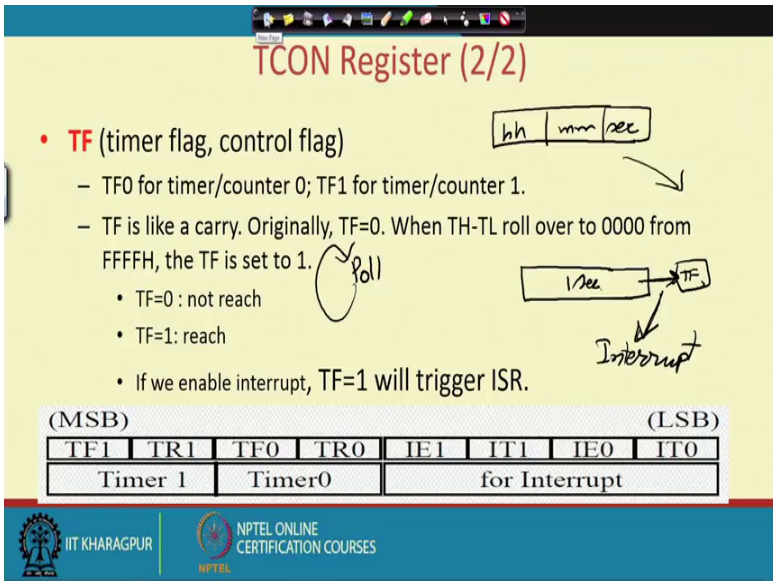
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Next interesting register, the that we have for this timer counter operation is the TCON register. So, TCON register. So, this is the timer control registered is TMOD. So, upper nibble where there is for timer counter, lower nibble is for interrupt. So, this is. So, this is the timer control register; so, TCON..

So, TMOD we have already seen. So, this is about the TCON register. So, this register you can divide it into 2 nibbles. So, this is the lower nibble. So, lower nibble we will not discuss now. So, they are for interrupts, for upper nibble is for the timer operation, and here you see those bits TR 0 ad TR 1. So, TR 0 is for running the timer counter 0 and TR 1 is for running the timer counter 1.

So, TR 0 if you set this bit to 0, then the timer will be stopped, and if you set the TR b to 1, then the timer will be started. So, previously we have seen that the; so, for soft controlled or software-controlled timers. So, you should program this TR 0 and TR 1 bits. So, those bits can be use those bits can be set by using this this by using sum instruction that will affect this TR 0 and TR 1 bits, and this this TF 0 and TF 1 also you have seen that is basically the overflow, in when the overflow occurs then this TF 0 and TF 1 will be set..

(Refer Slide Time: 04:43)


TCON Register (2/2)

- **TF** (timer flag, control flag)
  - TF0 for timer/counter 0; TF1 for timer/counter 1.
  - TF is like a carry. Originally, TF=0. When TH-TL roll over to 0000 from FFFFH, the TF is set to 1.
    - TF=0 : not reach
    - TF=1: reach
    - If we enable interrupt, TF=1 will trigger ISR.

(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

So, that TF or the timer flag. So, here TF 0 is for timer counter 0, TF 1 is for timer counter 1. So, it is like carry. So, initially TF is equal to 0, and then when this TH TL will rollover from all f to 0. So, if is a 16-bit timer, then when it is rolls over from 65 FFFFH to all 0 then the TF flag is set. So, TF equal to 0; so, we have not yet reached the final valuem and TF equal to 1 means we have reached the timer value..

So, what we can do while right generating as a delay. So, we can initialize the those timer with some value with some initial value, then start the timer by setting the corresponding TR bit and now continually monitor whether these TF bit is set or not, because once the TF bit is set means that time has passed. So, you have that time value has been used.

So, there are interrupts. So, we can. So, there are. So, we can do it in 2 ways. So, one is via this polling mechanism, another is via this interrupt mechanism. So, what I say is like this. So, if I have got this timer, now this timer when it workflows. So, it will set that TF flag. So, what you can do, you can either poll for poll this TF flag. So, whether it is 0 whether it is 1 or not, and you can continually do that. So, you can initialize the timer and then start it and continually poll the TF bit whether it has become 1 or not.

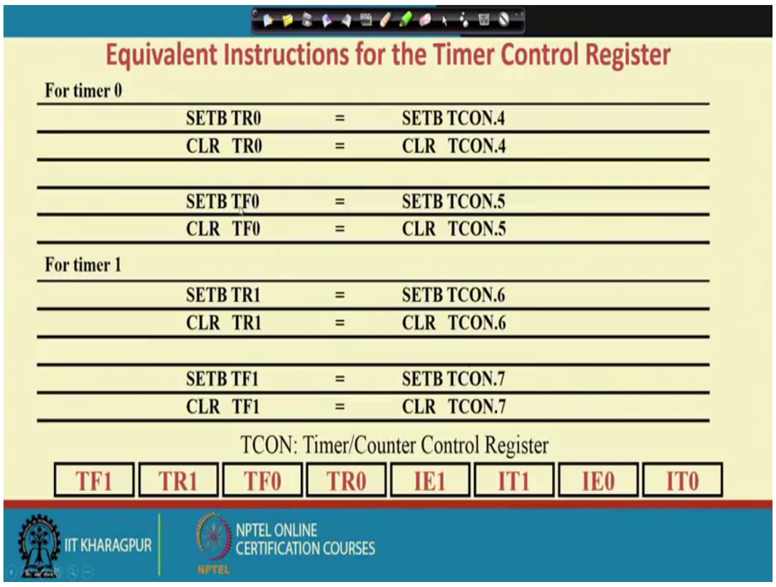
Other possibility is you can enable the timer interrupt. So, we see how to enable the timer interrupt while discussing about the interrupts, but once if this is enabled, then whenever this TF is overflow. So, this will generate and interrupt to the processor. So, the processor maybe doing something else, and it now it will be up now it will be interrupted. So, it will go to the timer service routine. So, this is typically useful when you are suppose designing a digital watch. So, in the digital watch. So, I have got this is hour part then minute part and second part fine.

So, the basic or the main program it may be continually getting those hour minute and second values from some memory location and displaying it, and side by side it needs to update the time for updating the time. So, every second the processor should update the time value, and how will it do it. So, it will start a timer for a value of 1 second, and whenever this timer overflows the interrupt is enabled..

So, you can have an ISR interrupt service routine for that purpose and then what that interrupt service routine will do, it will be updating the corresponding memory location for this hour minute second those bytes and naturally. So, this my display routine since it is reading from that location. So, it will be displaying the correct time value. So, this way we can use this timers for designing some real time clock..

So, this TF flags are that TF bits are useful that way. So, next we will look into the instructions, that are that can be that are required for this timer operation.

(Refer Slide Time: 08:09)



Equivalent Instructions for the Timer Control Register

For timer 0			
	SETB TR0	=	SETB TCON.4
	CLR TR0	=	CLR TCON.4
	SETB TF0	=	SETB TCON.5
	CLR TF0	=	CLR TCON.5
For timer 1			
	SETB TR1	=	SETB TCON.6
	CLR TR1	=	CLR TCON.6
	SETB TF1	=	SETB TCON.7
	CLR TF1	=	CLR TCON.7

TCON: Timer/Counter Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

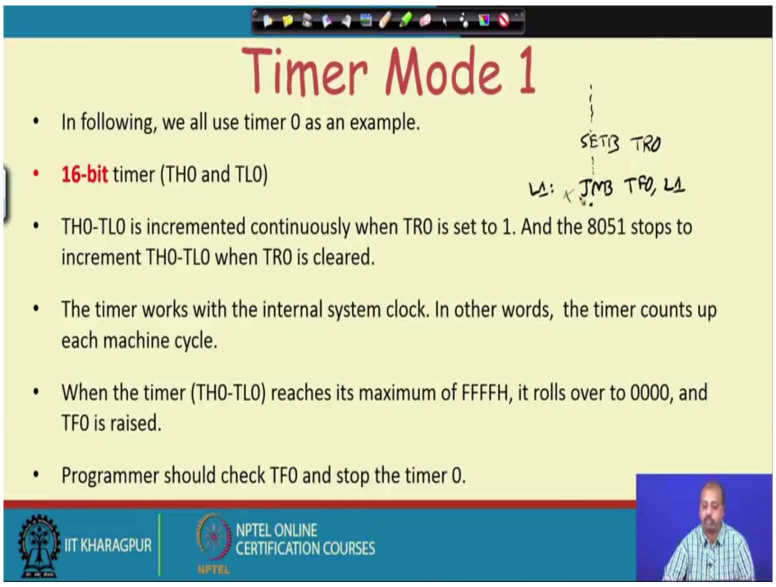
So, SETB TR 0. So, this is bit addressable this TCON register is bit addressable. So, you can say like SETB TR 0, or SETB TCON dot 4, or you can say clear TR 0. So, SETB TR

0 if you do then timer 0 will start, and if you are doing this set clear TR 0 then the timer 0 will stop. Similarly, we can have this SETB TF 0..

So, it will set this timer overflow flag to 0 and clear TF 0. So, also this SETB TF 0 is generally not required, but this clear TF 0 is required, because whenever you are sampling this line. So, if there is an overflow then this bit will be then this bit will be set by the system. So, in your ISR. So, you may like to reset that bit first otherwise it will be taken as another interrupt..

So, that way this clear bit is necessary, similarly for timer 1 we have got this SETB TR 1 and clear TR 1, and we have got this SETB TF 1, and clear TF 1 so, those things, and this 4 bits IE1 and IT 1 IT 0. So, this will come when you go into the interrupts..

(Refer Slide Time: 09:33)



Timer Mode 1

- In following, we all use timer 0 as an example.
- 16-bit timer (TH0 and TL0)
- TH0-TL0 is incremented continuously when TR0 is set to 1. And the 8051 stops to increment TH0-TL0 when TR0 is cleared.
- The timer works with the internal system clock. In other words, the timer counts up each machine cycle.
- When the timer (TH0-TL0) reaches its maximum of FFFFH, it rolls over to 0000, and TF0 is raised.
- Programmer should check TF0 and stop the timer 0.

SETB TR0

L1: JNB TF0, L1

So, next we will look into the timer mode 1. So, timer mode 1 is. So, it is 16-bit timer as we know. So, in our examples. So, we will consider only timer 0 and knowingfully well that the same thing will also be true for timer 1. So, this 16-bit timer. So, it has got TH 0 and TL 0 register pair. So, TH 0 TL 0 pair will be incremented continually, once this TR 0 is set to 1

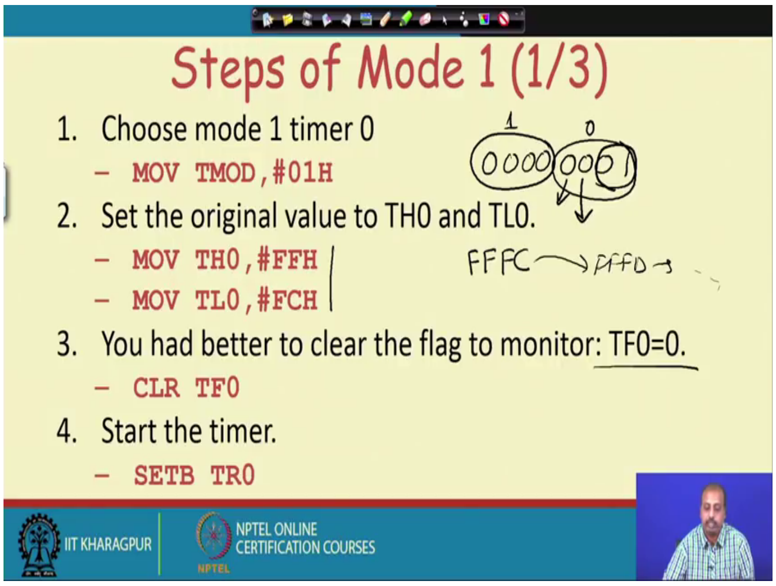
So, TR 0 being set to 1. So, they will be this value will be incremented continually, and it will stop when this TH 0 TL 0 pair will reach 0 or say this TR 0 has been is clear, like in a program after some time. So, you can stop this timer by see clearing the TR 0, though it has not overflown..

So, you can stop it in between by clearing the TR 0 bit, or otherwise this timer will continually update. So, it will be looking into the internal system clock. So, it will be updating itself, and then when it will be at the end of each machine cycle. So, it will be counting it up by 1, because that is the driving of the internal clock.

So, when it reaches the value FFFFH then it will roll over to 0 0 0 0, and then the TF 0 bit will be raised. So, that is the mode 1 operation of this timer, and programmer can check TF 0 and stop the timer 0. So now, it now the programmer are as I said that it may be a polling. So, programmer in a pole in a polling mode can continually check for this TF 0 a TF 0 bits. So, I can have an instruction like this that jump sorry, I can have an instruction like after setting the timer starting the timer. So, somewhere I have got this SETB TR 0. So, timer has started and after some time. So, I can if I just want to have a delay. So, you can say that jump on not bit TF 0 L 1 where L 1 is this this place itself.

So, this is the polling code. So, I am checking the bit TF 0 and if it is not yet set. So, I am just looking at this point. So, this way. So, whenever this bit will be is set. So, it will be coming to the this check will be false. So, it will be coming to the next position, this way we can have this type of polling implemented in our code. Next so, how to use this mode 1.

(Refer Slide Time: 12:32)



Steps of Mode 1 (1/3)

1. Choose mode 1 timer 0
   - MOV TMOD,#01H
2. Set the original value to TH0 and TL0.
   - MOV TH0,#FFH
   - MOV TL0,#FCH
3. You had better to clear the flag to monitor: TF0=0.
   - CLR TF0
4. Start the timer.
   - SETB TR0

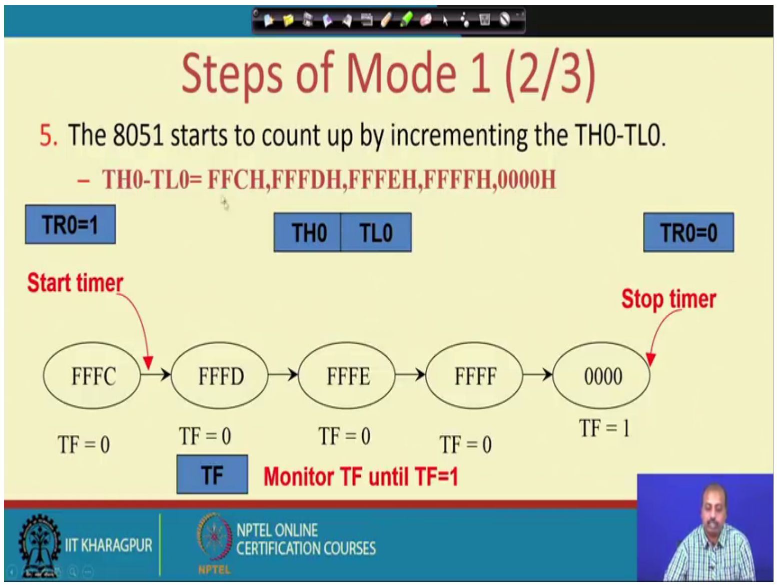
So, first we have to choose mod 1 for timer 0, and for that matter. So, we set the controlled over for in the TMOD register to be 0 to be 0 1 H. So, 0 1 H means the first 4 bits will be 0, first 4 bits will be 0, and the next 4 bits will be 0 0 0 1.

So, this part is for timer 1. So, that is not our concern now and this part is for timer 0 out of that this is the gate so; that means, I am driving for from internal source the clock is internal source. So, this 0 will mean that it is a timer operation and then these 2 bits. So, they will identify that I am operating in mode 1 that is 16-bit timer mode the next important thing is to load this TH 0 TL 0 pair..

So, this pair I am loading as FF FC. So, the count value that I have loaded is FFFC, and it is better that we clear this TF 0 bit. So, for the safety because previously maybe this TF 0 was set, and it was not cleared before this timer had started. So, it is to be on the safe side we should clear the TF 0 bit. So, we clear the TF 0 bit here and then we start set bit TR 0. So, SETB TR 0. So, it will start the timer now. So, from FFFC. So, it will go to FFFD then FFFE. So, it will go like that. So, when it comes to 0, then there is an overflow. So, that will be detected..

So, this is the thing. So, 8051 will start counting like this.

(Refer Slide Time: 14:23)



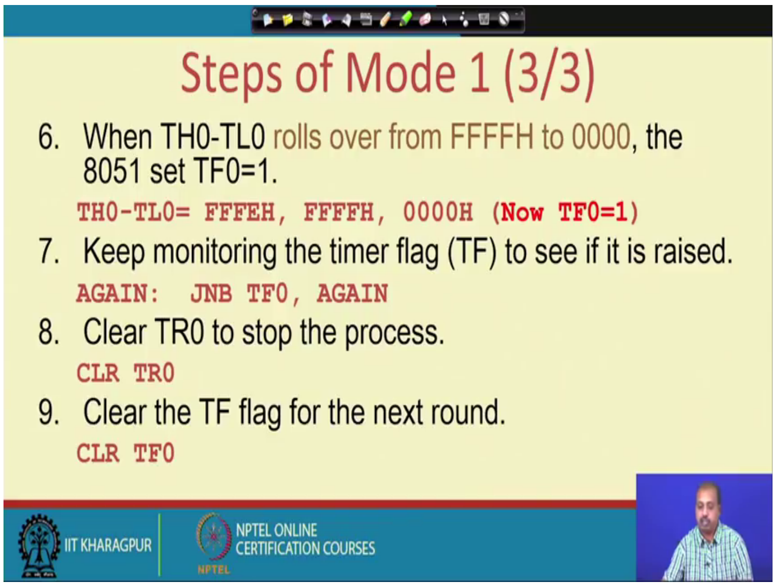
Steps of Mode 1 (2/3)

5. The 8051 starts to count up by incrementing the TH0-TL0.
   - TH0-TL0= FFFCH,FFFDH,FFFEH,FFFFH,0000H

So, it will be FFFC, FFFD, FFFE and FFFF then this 0 0. So, when it goes to this 1. So, this FFFF to 0 0 0 0 at that point. So, if you monitor this TF 1 this TF 1 bit will be set to

1. So, previously this TF g bit was 0, this TF this this all TF 0 bit. So, this is not 0 is not explicitly mentioned here. So, this is the TF 0 bit. So, that is set to 0, and when this overflow occurs then it will be coming. So, we can monitor this situation whether this TF 0 is becoming 1 or not, and when this TF 0 becomes 1. So, we should stop the timer. So, you can stop this TR 0, you can reset this TR 0 bit accordingly.

(Refer Slide Time: 15:19)



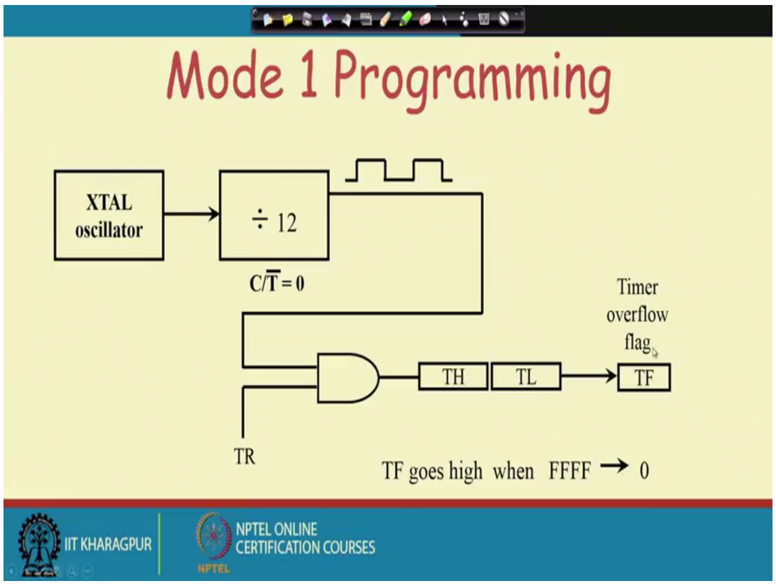
Steps of Mode 1 (3/3)

6. When TH0-TL0 rolls over from FFFFH to 0000, the 8051 set TF0=1.
   `TH0-TL0= FFFEH, FFFFH, 0000H (Now TF0=1)`
7. Keep monitoring the timer flag (TF) to see if it is raised.
   `AGAIN:  JNB TF0, AGAIN`
8. Clear TR0 to stop the process.
   `CLR TR0`
9. Clear the TF flag for the next round.
   `CLR TF0`

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, when this TH 0 TL 0 rolls over from FFFF to 0 0 0 0, the 8051 will set TF 0 bit to 1. So now, this we can we can do it as I was telling that we can have a monitoring like this, that jump on not bit this TF TF 0 again. So, we can have something like that. So, that will mean that we are monitoring thus TR 0 bit, and as long as it is 0 so, it will be looking here then we clear the TR 0 bit to stop stop the timer operation, and then we clear the TF flag to so, that next time when you want to operate the timer, it should not create any problem. So, this way we can use this timer 16-bit timer in mode 1.

(Refer Slide Time: 16:13)



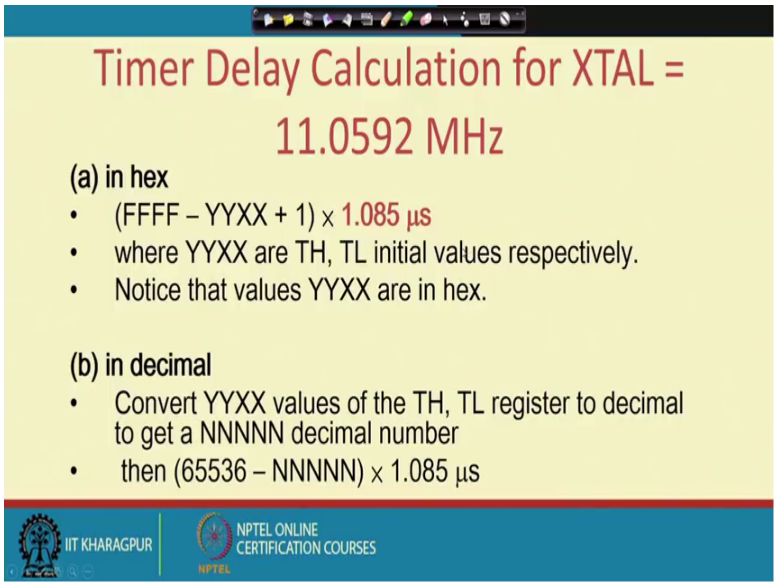
Mode 1 Programming

So, the operation is like this the crystal frequency. So, that is divided by this 12. So, this is in c by T equal to 0. So, this 12 cycles in case of 8051; 12 clock cycles constitute 1 machine cycle, and this timer value is updated per machine cycle so, that is why so, it is the internally it is divided by 12. So, this division is internal..

So, it is internally divided by 12 so, this type of a clock is generated, and then this it is fate to an n gates, this is all logical. So, whether it how is it physically implemented. So, that is not known, but logically you can think about it as if this TR bit is ended with the clock, and when this TR bit is 1 then only this this clock will be allowed to reach this timer value, and this timer value will start uptime upcounting.

So, from FF from whatever be the value. So, it will go on increasing, and then with it rolls over from FFFF to 0, then this timer overflow flag f TF will be set. So, how to calculate the delay.

(Refer Slide Time: 17:26)



Timer Delay Calculation for XTAL = 11.0592 MHz

(a) in hex

- (FFFF – YYXX + 1) x 1.085 μs
- where YYXX are TH, TL initial values respectively.
- Notice that values YYXX are in hex.

(b) in decimal

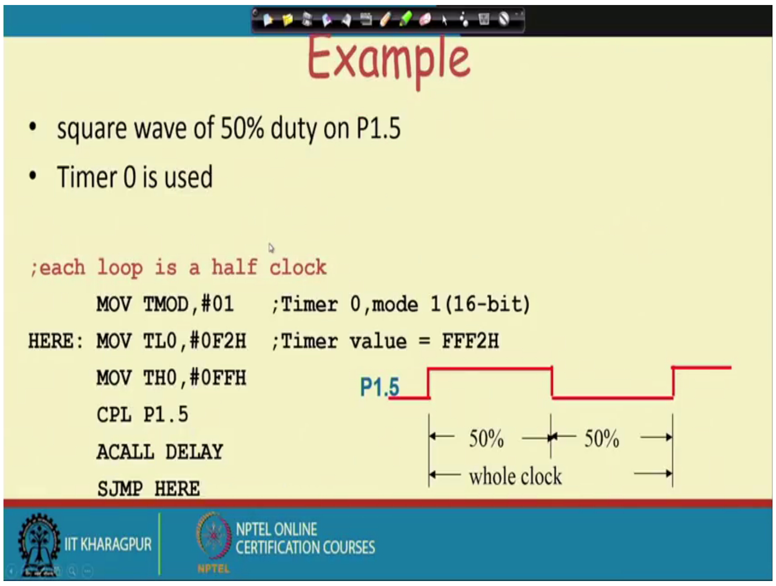
- Convert YYXX values of the TH, TL register to decimal to get a NNNNN decimal number
- then (65536 – NNNNN) x 1.085 μs

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, it depends on the crystal frequency of course, but how can I do this? For example, suppose the value that I have initialized with this this TH and TL register pair is YYXX hex. So, if it is YYXX hex you know that if. So, how many cycles will it take to go to FFFF. So, it is FFFF minus YYXX plus 1. So, after so many cycles overflow will occur, and if this is 11.0592 is the crystal frequency. So, that turns out to be 1.085 microsecond.

So, that way I can say that it is so, this is this has to be so, this is divided by 12 first of all, if we look into this previous slide. So, this this crystal oscillator is divided by 12 and that is fed as clock here. So, here also. So, this divided by 12 and then if you convert it into time. So, 1 by f. So, that turns out to be 1.085 microsecond or in decimal you can just. So, same thing instead of taking FFFF FF plus 1. So, you can say simply write like 65536, minus this YYXX in decimal maybe say NNNNN. So, this is multiplied by 1.085 microseconds. So, this way given this time value the timer initial value. So, we can calculate what is the actual delay that is produced.

(Refer Slide Time: 18:57)



Example

- square wave of 50% duty on P1.5
- Timer 0 is used

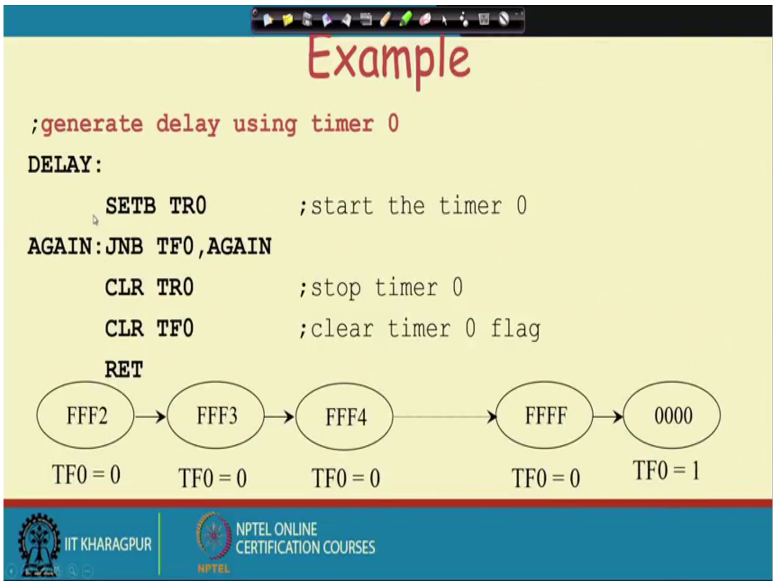
```
;each loop is a half clock
      MOV TMOD,#01    ;Timer 0,mode 1(16-bit)
HERE: MOV TL0,#0F2H   ;Timer value = FFF2H
      MOV TH0,#0FFH
      CPL P1.5
      ACALL DELAY
      SJMP HERE
```

P1.5
50%
50%
whole clock
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Suppose we want to design a square wave with 50 percent duty cycle on 4 bit 1.5. So, we have seen previously this example, but there we were using that soft delays, now we are now here we will be using this timer delays. So, we are using this timer 0 mode 1 16 bit then move TL 0 f 2 h. So, and this TH 0 0 FFH. So, this timer value that is loaded is FFFFF 2..

So, this is value that is loaded, now we compliment p 1.5. So, that way we whatever be the previous value of this port. So, that gets complimented then we give a call delay. So, the delay routine is called. So, we will see that delay routine and then SJMP here, if it will come back to this point. So, again this timer will be loaded and the value will be complimented.

(Refer Slide Time: 19:54)



Example

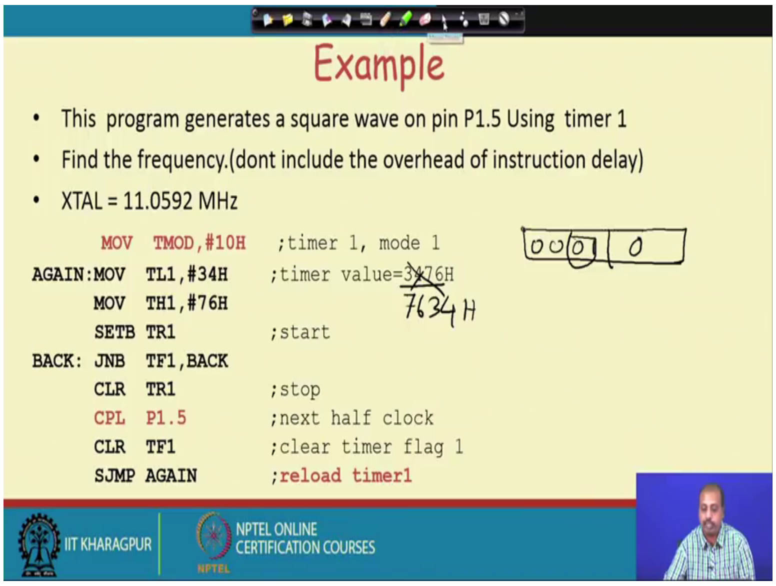
```
;generate delay using timer 0
DELAY:
        SETB TR0        ;start the timer 0
AGAIN:JNB TF0,AGAIN
        CLR TR0         ;stop timer 0
        CLR TF0         ;clear timer 0 flag
        RET
```

So, the delay routine is like this so, we set bit this TR 0 this timer is started. So, previously the delay routine was doing some registered decrementing jump not 0 etcetera. So, here instead of that we are starting the timer, and we are now monitoring the TF 0 bit. So, this JNB TF 0 again. So, this will be continually looping here, till this TF 0 bit is set to 1. So, when this t 0 is set to 1, at that time this this delay has been seen. So, it is clear TR 0 and clear TF 0. So, we stop the timer 0, and it will clear the timer 0 flag and it will return. So, this program it will be doing that thing like say it will be doing this operation.

So, after that again. So, after this after we have returned from this delay routine. So, it is basically this jumping to this point here, and here again we are initializing the timer to TL 0 and TH 0 those 2 registers were initializing 2 values. So, this value is same for both time high and time low..

So, you get this value as the as a 50 percent duty cycle. So, you can also very easily calculate what is the delay that is produced, because the count value that you have is the initial value is FFF 2. So, using the previous formula that we have got here. So, you can calculate what is the ontime and offtime. So, accordingly you can say that. So, so much of what will be the frequency of this square wave that is generated. So, that can be calculated very easily.

(Refer Slide Time: 21:41)



Example

- This program generates a square wave on pin P1.5 Using timer 1
- Find the frequency.(dont include the overhead of instruction delay)
- XTAL = 11.0592 MHz

```
        MOV  TMOD,#10H    ;timer 1, mode 1
AGAIN:MOV  TL1,#34H     ;timer value=3476H
       MOV  TH1,#76H
       SETB TR1         ;start
BACK: JNB  TF1,BACK
       CLR  TR1         ;stop
       CPL  P1.5        ;next half clock
       CLR  TF1         ;clear timer flag 1
       SJMP AGAIN       ;reload timer1
```

7634 H

0 0 0 1 0

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

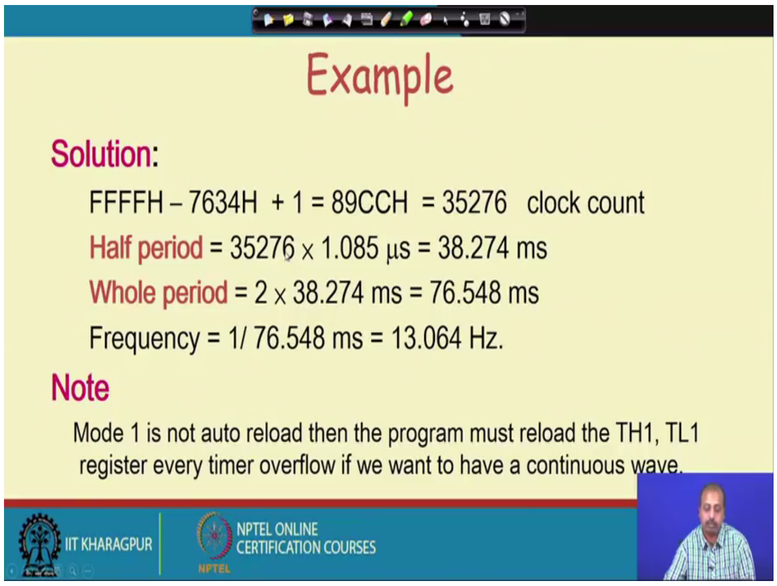
Next so, we try to find the frequency. So, this program. So, this is generating a square wave on p 1.5 using timer 1, and we want to find the frequency, now so, first off, all this TMOD setting. So, TMOD setting second nibble is 0..

So, this should be TH 1 it is not if you if you want to get this 3 4 7 6 here. So, this is not 3 4 7 6 sorry. So, this is not 3 4 7 6. So, this is 7 6 3 4, because this TH register has got 7 6 and this TL register has got there 3 4. So, it is 7 6 3 4. So, we are initializing this TH TL pair, then set bit TR 1. So, we are starting the timer 1 at this point, and then jump on not bit TF 1 back. So, this is basically that polling of that plane TF 1.

Then we are clearing TR 1. So, TR 1. So, this timer is this timer is stopped, then we are complimenting the bit then we are clearing the TF 1 bit. So, TF 1 bit is cleared so, this timer is now that previously whatever this TF 1 was set. So, that is cleared at this point, and then SJMP again. So, it goes to here and again the timer is loaded. So, this is again 50 percent duty cycle case, but the value that we have loaded is 7 6 3 4 hex..

So, question is to find the frequency and it just has to simplify the problem, we do not include the delays of these individual instructions. So, to get a more accurate thing like you have to take into account this individual instruction delays also, but they are they may not be very high compared to the delay that is produced by the timer. So, they can be neglected for our discussion..

(Refer Slide Time: 24:59)



Example

Solution:

FFFFH – 7634H + 1 = 89CCH = 35276 clock count

Half period = 35276 x 1.085 µs = 38.274 ms

Whole period = 2 x 38.274 ms = 76.548 ms

Frequency = 1/ 76.548 ms = 13.064 Hz.

Note

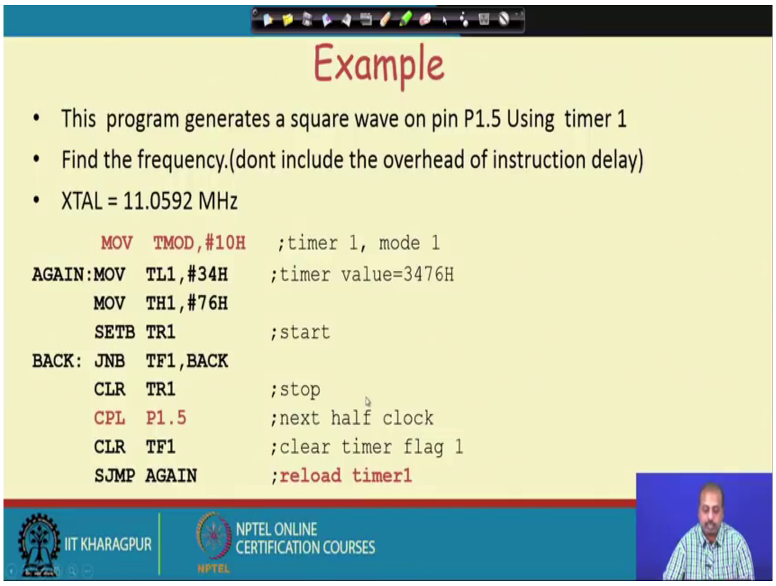
Mode 1 is not auto reload then the program must reload the TH1, TL1 register every timer overflow if we want to have a continuous wave.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, what is that so, we have loaded this initial value of 7 6 3 4. So, our total clock count is FFFF minus 7 6 3 4 plus 1. So, that is. So, that is 8 9 CC the 3 5 2 7 6 clock time clock count. So, this half period. So, this is. So, since this delay is used for both on part and off part. So, we can say that for each of the parts. So, delay is 38.274 millisecond, the whole delay is the whole-time period of the signal is 2 into 38.274 millisecond that is 76.548 millisecond. So, frequency of the square wave that is generated is 1 upon 76.548 millisecond, it is 13.064 hour.

So, mode 1 is. So, this is not auto reload. So, so we need to reload the TH 1 and TL 1 registers separately, and then that then only we can this TH 1 and TL 1 registers, every time you need to restart the counter restart the timer, you have to reload this TH 1 and TL 1 pair, the same is true for this timer 0; however, if you are looking for this auto reload mode. So, you have to use the mode 2..

(Refer Slide Time: 26:32)



Example

- This program generates a square wave on pin P1.5 Using timer 1
- Find the frequency.(dont include the overhead of instruction delay)
- XTAL = 11.0592 MHz

```
        MOV  TMOD,#10H   ;timer 1, mode 1
AGAIN:MOV  TL1,#34H     ;timer value=3476H
      MOV  TH1,#76H
      SETB TR1          ;start
BACK: JNB  TF1,BACK
      CLR  TR1          ;stop
      CPL  P1.5         ;next half clock
      CLR  TF1          ;clear timer flag 1
      SJMP AGAIN        ;reload timer1
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, in in that case many time this reloading here as we have done in this case. So, every time after producing something again we are loading these values. So, that will not be required. So, we will see that in mode 2 discussion.