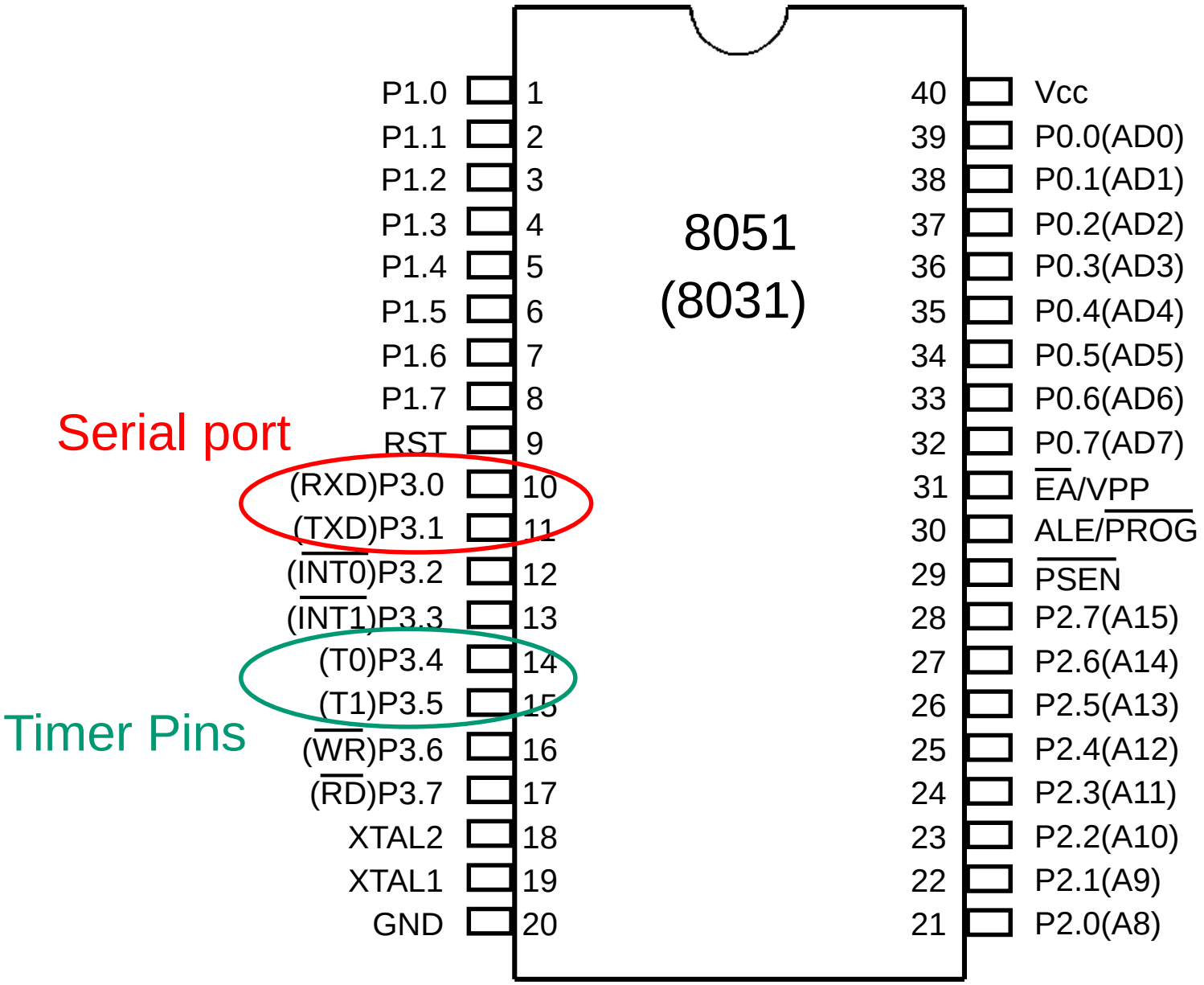
Module 6

8051 Timers and Serial Port

8051 Pin Diagram



Inside Architecture of 8051

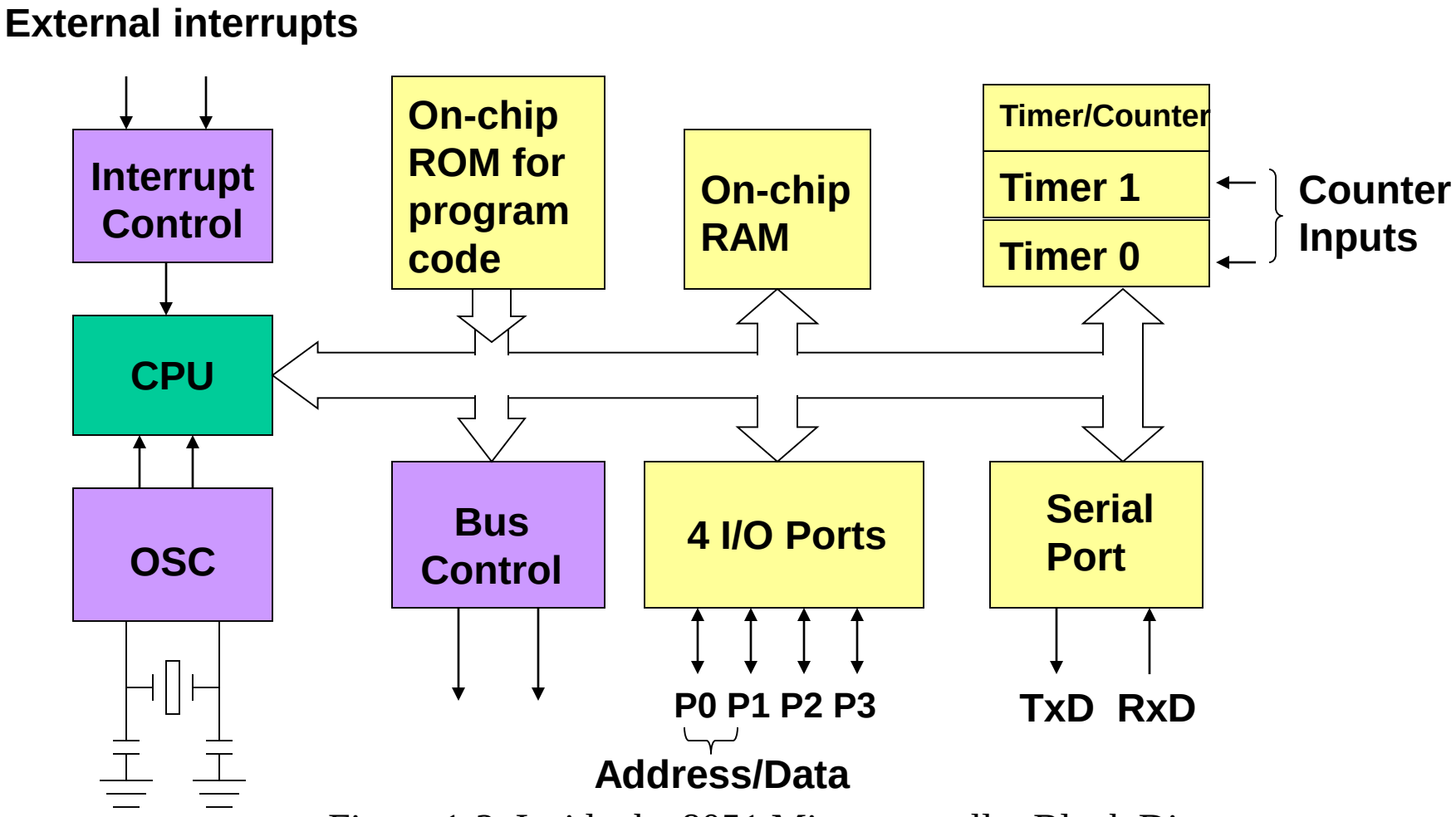
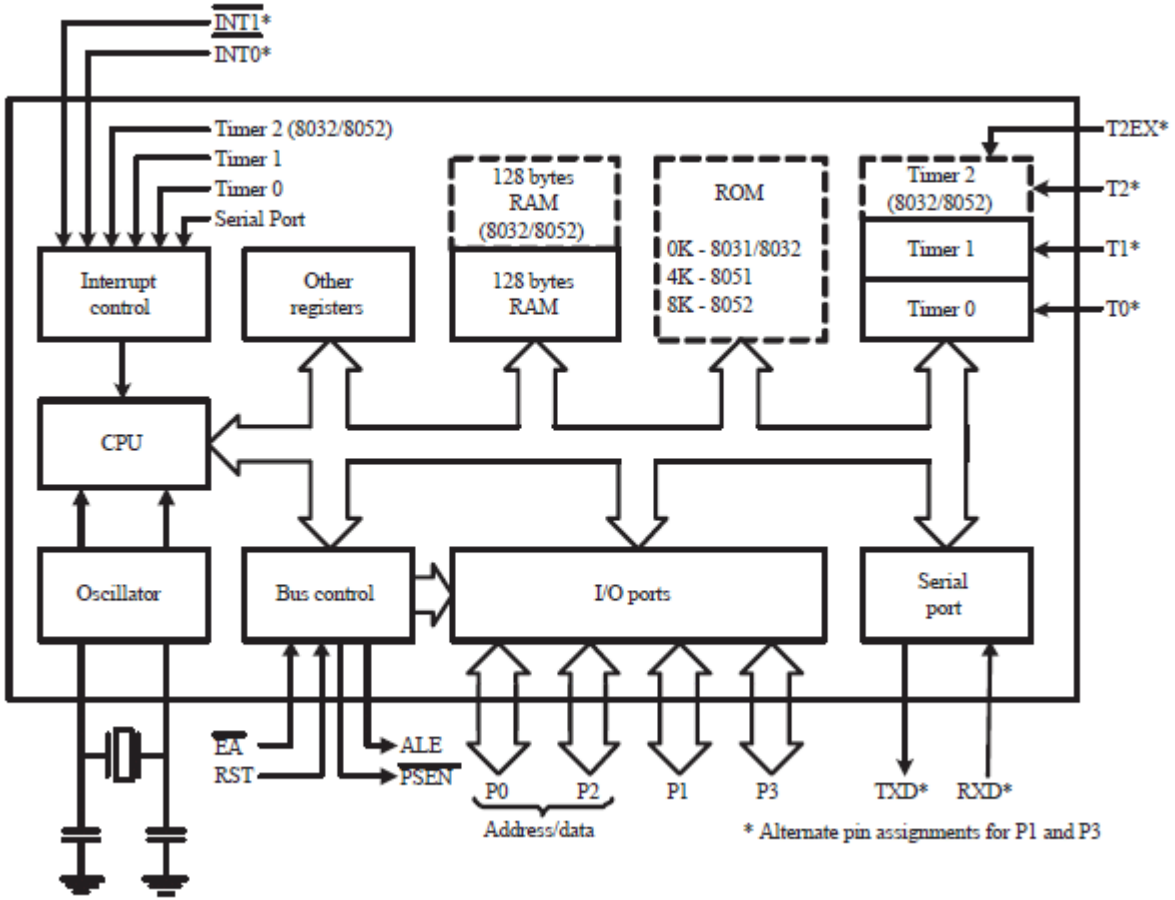


Figure 1-2. Inside the 8051 Microcontroller Block Diagram

Inside Architecture of 8051



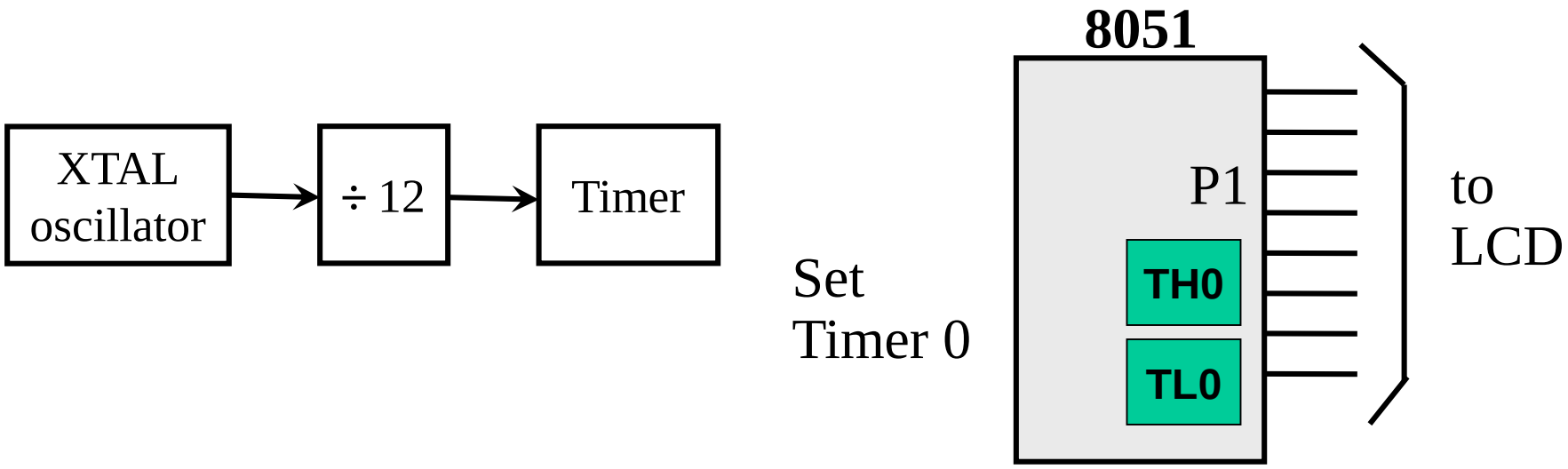
Timers /Counters

- The 8051 has 2 timers/counters: timer/counter 0 and timer/counter 1. They can be used as

1. The **timer** is used as a time delay generator.
   - The clock source is the **internal** crystal frequency of the 8051.
2. An event **counter**.
   - **External input** from input pin to count the number of events on registers.
   - These clock pulses could represent the number of people passing through an entrance, or the number of wheel rotations, or any other event that can be converted to pulses.

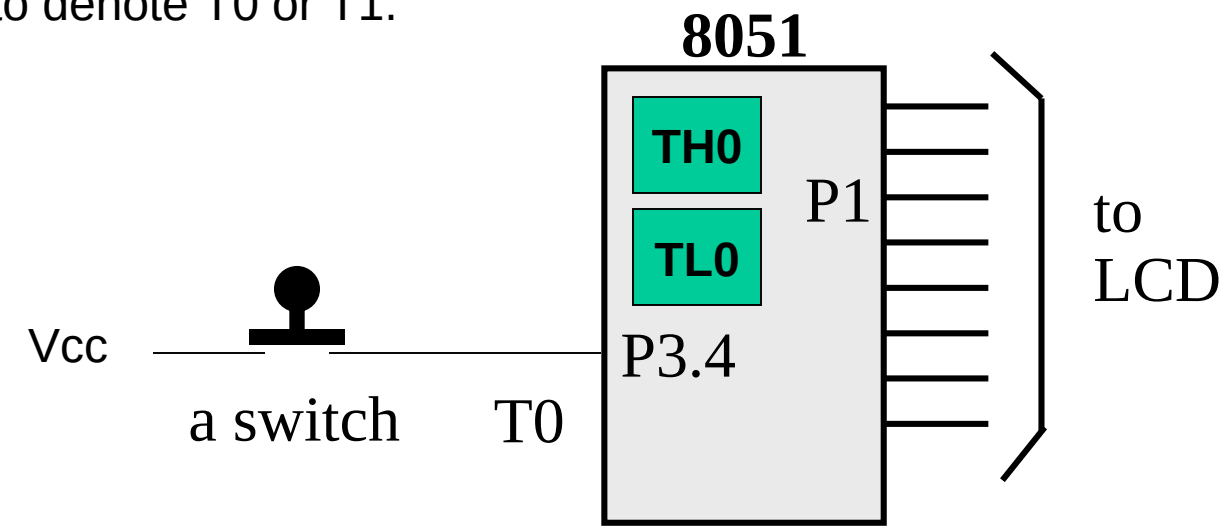
Timer

- Set the initial value of registers
- Start the timer and then the 8051 counts up.
- When the registers equal to 0, the 8051 sets a bit to denote time out
- 8051 timers use 1/12 of XTAL frequency, regardless of machine cycle.



Counter

- Count the number of events
  - Show the number of events (1 to 0 transition) on registers
  - External input from T0 input pin (P3.4) for Counter 0
  - External input from T1 input pin (P3.5) for Counter 1
  - The external input is sampled at each machine cycle (S5P2)
  - The new value appears in timer registers during S3P1 of the cycle following the one in which the transition is detected
  - We use Tx to denote T0 or T1.



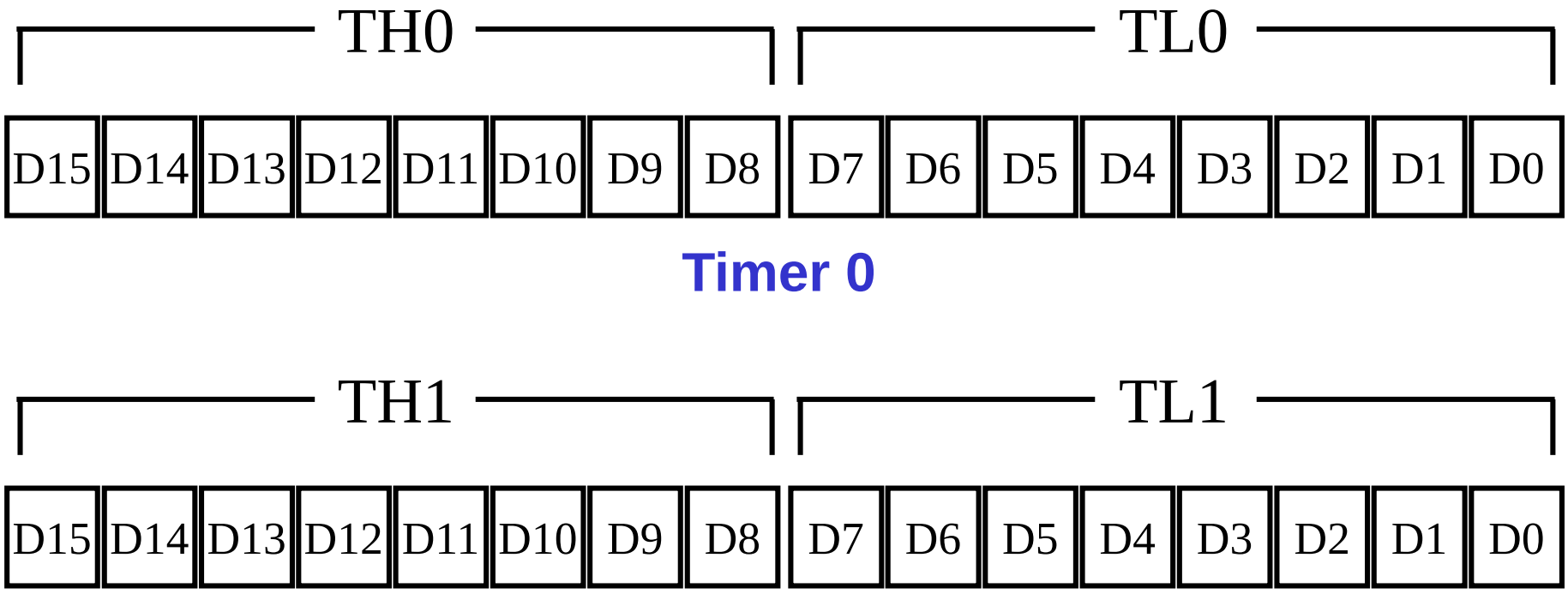
Registers Used in Timer/Counter

- TH0, TL0, TH1, TL1 (Timer 0 and Timer 1 registers)
- TMOD (Timer mode register)
- TCON (Timer control register)
- Since 8052 has 3 timers/counters, the formats of these control registers are different.
  - T2CON (Timer 2 control register), TH2 and TL2 used for 8052 only.

Basic Registers of the Timer

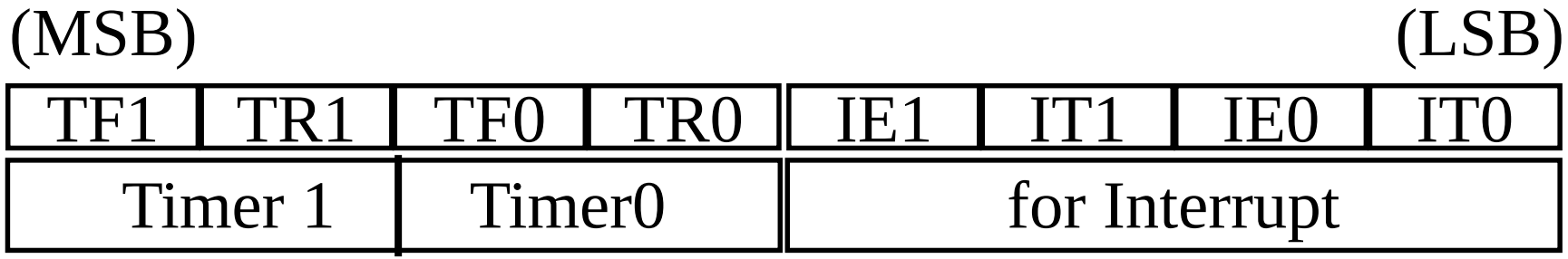
- Both Timer 0 and Timer 1 are 16 bits wide.
  - These registers store
    - the time delay as a timer
    - the number of events as a counter
  - Timer 0: **TH0** & **TL0**
    - Timer 0 high byte, timer 0 low byte
  - Timer 1: **TH1** & **TL1**
    - Timer 1 high byte, timer 1 low byte
  - Each 16-bit timer can be accessed as two separate registers of low byte and high byte.

Timer Registers



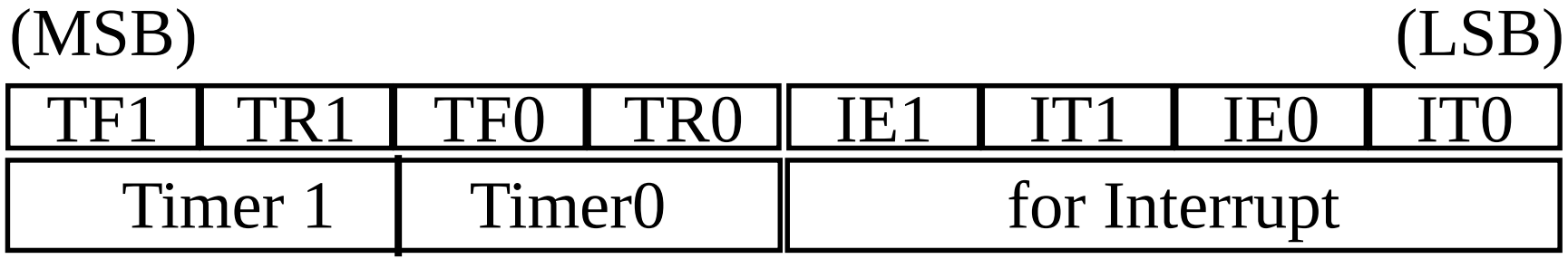
TCON Register

- Timer control register: **TCON**
  - 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)



TCON Register

- **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 rolls over to 0000 from FFFFH, the TF is set to 1.
    - TF=0 : not reached
    - TF=1: reached
    - If we enable interrupt, TF=1 will trigger ISR.



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

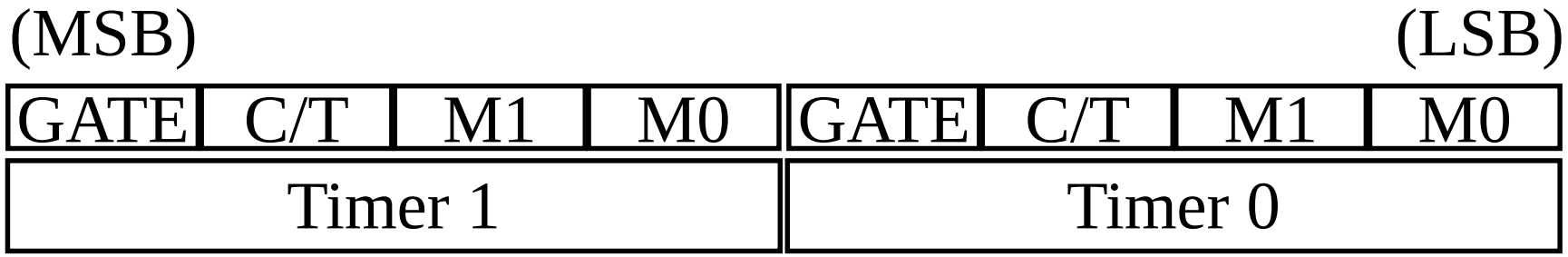
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TMOD Register

- Timer mode register: **TMOD**

```
MOV TMOD,#21H
```

  - An 8-bit register
  - Set the usage mode for two timers
    - Set lower 4 bits for Timer 0 (Set to 0000 if not used)
    - Set upper 4 bits for Timer 1 (Set to 0000 if not used)
  - Not bit-addressable



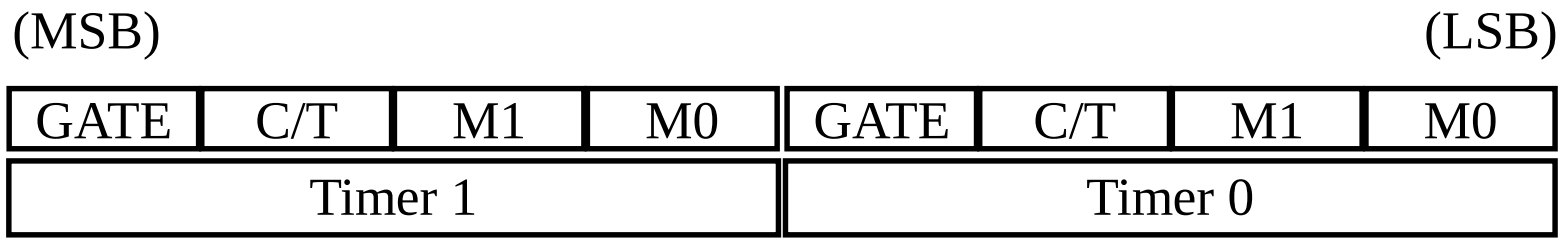
TMOD Register

GATE Gating control when set. Timer (or counter) 0/1 is enabled only while the P3.2/P3.3 pin is high and the TRx control bit is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T Select Timer or Counter. Cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).

M1 Mode bit 1

M0 Mode bit 0



Counter/Timer $C/\overline{T}$

- This bit is used to decide whether the timer is used as a delay generator or an event counter.
- $C/\overline{T} = 0$: timer
- $C/\overline{T} = 1$: counter

Gate

- Every timer has a mean of starting and stopping.
  - GATE=0
    - **Internal** control
    - The start and stop of the timer are controlled by **software.**
    - Set/clear the TR for start/stop timer.
  - GATE=1
    - **External** control
    - The hardware way of starting and stopping the timer by **software *and* an external source**.
    - Timer/counter is enabled only while the INT pin is high and the TR control pin is set (TR).
    - INT0: P3.2, pin 12; INT1: P3.3, pin 13.

M1, M0

- M0 and M1 select the timer mode for timers 0 & 1.

M1	M0	Mode	Operating Mode
0	0	0	13-bit timer mode (compatibility to 8048) 8-bit THx + 5-bit TLx (x= 0 or 1)
0	**1**	**1**	**16-bit timer mode** **8-bit THx + 8-bit TLx**
1	**0**	**2**	**8-bit auto reload** **8-bit auto reload timer/counter;** **THx holds a value which is to be reloaded into** **TLx each time it overflows.**
1	1	3	Split timer mode

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:

TMOD= 0000 0010 Timer 1 is not used.

Timer 0, **mode 2**,

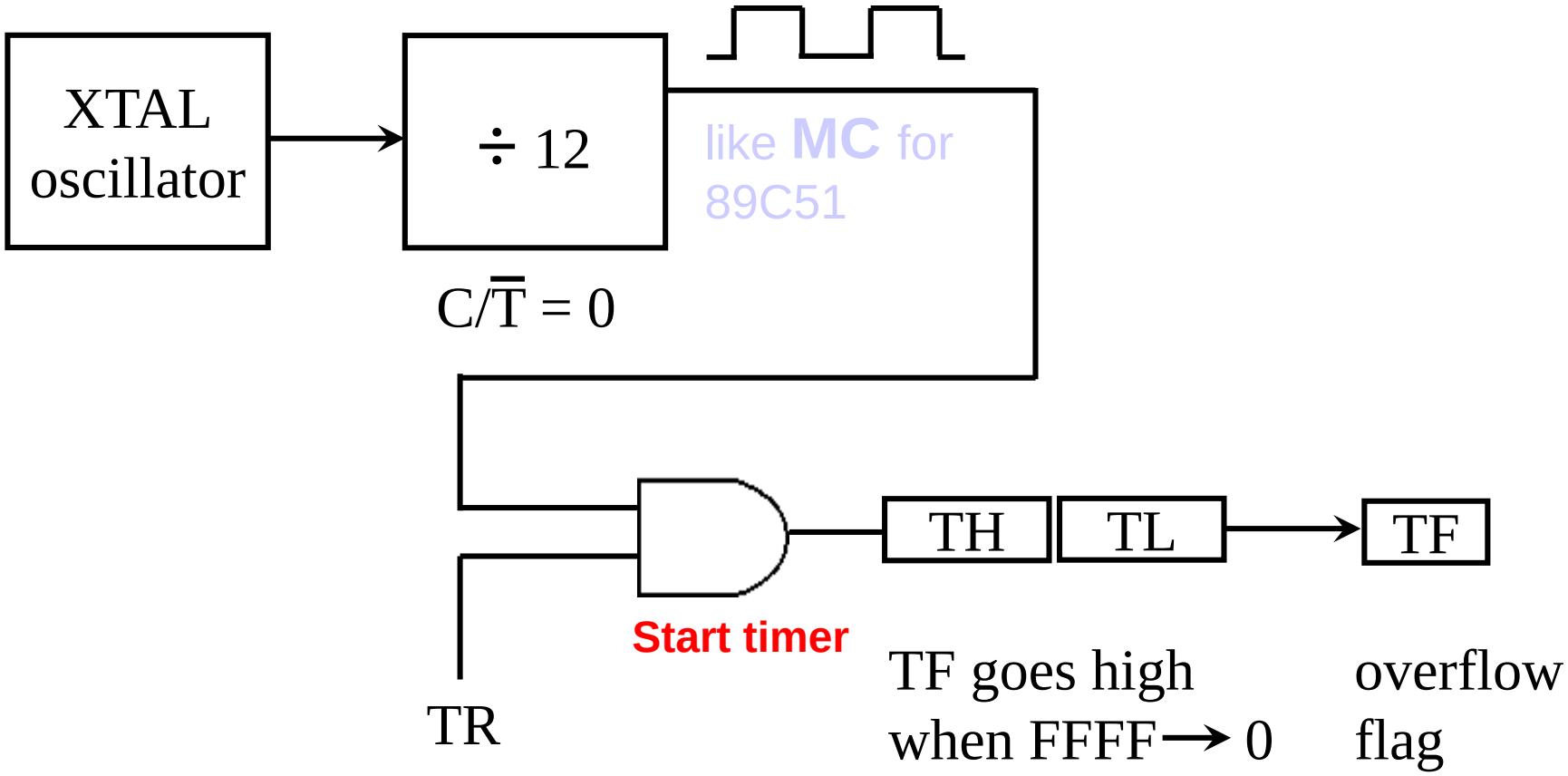
C/T = 0 to use XTAL clock source timer

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

Timer Mode 1

- In following, we will 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 every 12 clocks from XTAL.
- 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.

Mode 1 Programming

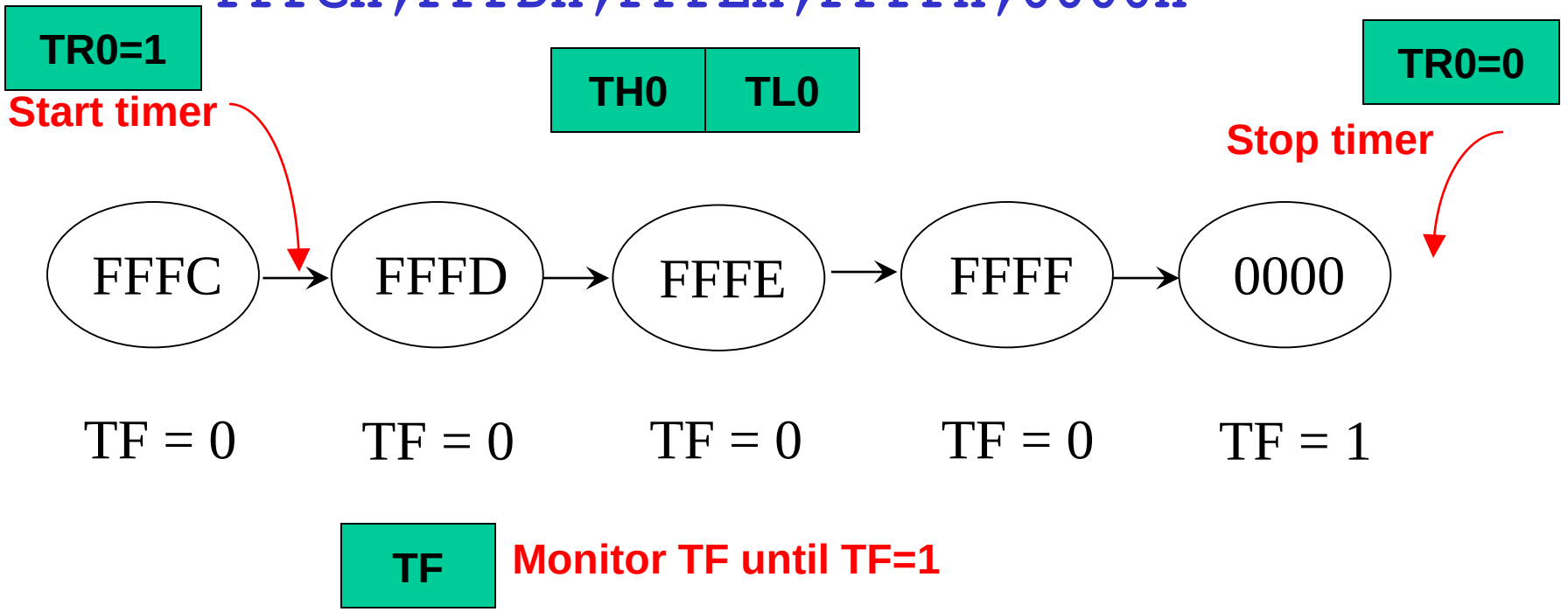


Steps of Mode 1

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

Steps of Mode 1

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



Steps of Mode 1

6. When TH0-TL0 rolls over from FFFFH to 0000, the 8051 sets 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**

Initial Count Values

- The initial count value = FFFC.
- The number of counts = FFFFH-FFFCH+1 = 4
  - we add one to 3 because of the extra clock needed when it rolls over from FFFF to 0 and raises the TF flag.
- The delay = 4 MCs for 89C51
- If MC=1.085 μs, then the delay = 4.34 μs

(a) in hex	(b) in decimal
(FFFF - YYXX + 1) × 1.085 μs where YYXX are TH, TL initial values respectively. Notice that values YYXX are in hex.	Convert YYXX values of the TH,TL register to decimal to get a NNNNN decimal number, then(65536 - NNNNN) × 1.085 μs

Counter Mode 1

- **16-bit** counter (TH0 and TL0)
- TH0-TL0 is incremented when TR0 is set to 1 **and** an external pulse (1-to-zero transition on T0) occurs.
- When the counter (TH0-TL0) reaches its maximum of FFFFH, it rolls over to 0000, and TF0 is raised.
- Programmers should monitor TF0 continuously and stop the counter 0.
- Programmers can set the initial value of TH0-TL0 and let TF0 as an indicator to show a special condition. (ex: 100 people have come).

Example

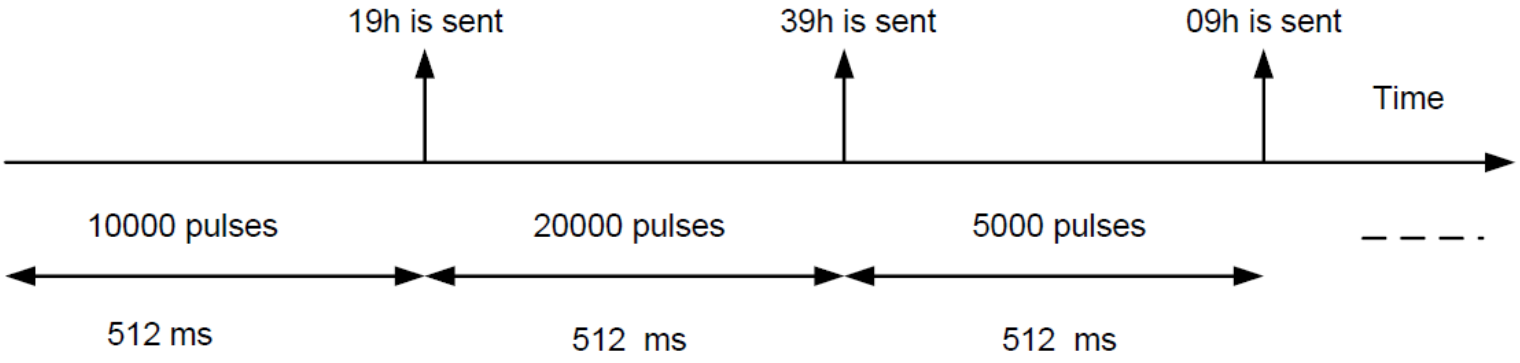
- Write a delay subroutine using Timer 0 in Mode 1 that generates a delay of
  - 40 ms
  - 4 seconds
- Assume CF=12 MHz.

Exam Question

Q4) [40 points] For an 8052 clocked at CF=1.2MHz, write a main program that will continuously listen to the timer 0 pin (T0 pin) for a waveform with frequency strictly bounded above by 100 kHz and count the number of pulses, i.e., 1 to 0 transitions, occuring at the T0 pin periodically with a period of T=512 ms. The main program will continuously send the frequency information in units of kHz as a packed BCD number to P1. For example if we count N=10000 pulses in a duration of T=512 ms, then the frequency f would be the integer part (whole part) of the ratio

$$f = \frac{10000}{512} kHz \approx 19kHz .$$

As a result, 19h should be sent. The following figure illustrates the operation of the system. **Hint: Use rotation for division by a power of two. Use timer 0 as event counter in Mode 1. Use timer 1 as timer in Mode 1. Send the frequency information every time the timer 1 overflows. Do not use the HIGH and LOW directives. All timer values are 00h at powerup.**



MAIN:	MOV TMOD,# 0001 0101b	; 15h
BACK:	MOV TH1,# 38h	; 1 TC=10 μs
	MOV TL1, #00h	; 51200 TCs needed
	SETB TR0	;run the timers here
	SETB TR1	
	JNB TF1,$	; wait for 51200 TC
	CLR TR0	
	CLR TR1	
	MOV A,TH0	; after reading the timer, divide by 512. Think of an elegant way. A=27h (example)
	RR A	; A = 13h for the example will have to make it 19h
	MOV B,#10	; convert to packed BCD
	DIV AB	;A=1,B=9 (example)
	SWAP A	;A=10h
	ADD A,B	;A=19h
	MOV P1,A	
	CLR TF1	; prepare
	MOV TH0,#0	; prepare for next counting
	MOV TL0,#0	
	SJMP BACK	

Timer Mode 2

- 8-bit timer.
  - TL0 is incremented (or roll over) continuously when TR0=1.
- Auto-reloading
  - TH0 is loaded into TL0 automatically when TL0=00H.
  - TH0 is not changed.
  - You need to clear TF0 after TL0 rolls over.
- In the following example, we want to generate a delay with 200 MCs on timer 0.
- See Examples 9-14 to 9-16

Steps of Mode 2

1. Choose mode 2 timer 0
   - `MOV TMOD,#02H`
2. Set the initial value to TH0 and TL0.
   - `MOV TH0,#38H`
   - `MOV TL0,#38H`
3. Clear the flag to TF0=0.
   - `CLR TF0`
4. Start the timer.
   - `SETB TR0`

Steps of Mode 2

5. The 8051 starts to count up by incrementing the TL0.
   - **TL0= 38H, 39H, 3AH,....**
6. When TL0 rolls over from FFH to 00, the 8051 sets TF0=1. Also, TL0 is reloaded automatically with the value kept by the TH0.
   - **TL0= FEH, FFH, 00H (Now TF0=1)**
   - The 8051 auto reloads **TL0=TH0=38H.**
   - Go to Step 6 (i.e., TL0 is incrementing continuously).

- Note that we must clear TF0 when TL0 rolls over. Thus, we can monitor TF0 in next process.
- Clear TR0 to stop the process.

Counter Mode 2

- 8-bit **counter**.
  - TL0 is incremented if TR0=1 and external pulse occurs.
- Auto-reloading
  - TH0 is loaded into TL0 when TF0=1.
  - It allows only values of 00 to FFH to be loaded into TH0.
  - You need to clear TF0 after TL0 rolls over.

Example

- Write a main program using Mode 2 that will continuosly generate a binary waveform with

$$T_{ON} = 100\mu s,\ T_{OFF} = 400\mu s,\ \text{CF=12 MHz}$$

Reading a Timer on the Fly

- Since two timer registers must be read, a phase error may occur if the low byte overflows into the high byte between the two read operations
- A value may be read that never existed
- Ex: Read THTL into R1R0

  TH:00 , TL:FF
  - First Read TL into R0, then read TH into R1:

    R0->FF , TL is incremented R1->01

    R1R0: 01FF
  - First read TH into R1, then read TL into R0

    R1->00, TL is incremented R0->00

    R1R0: 0000

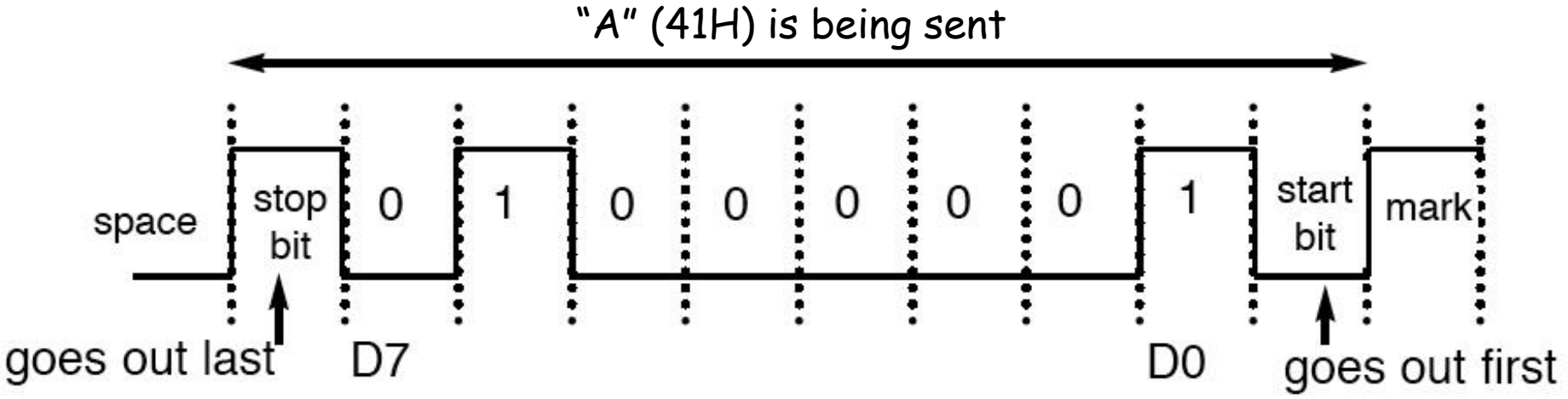
Reading a Timer on the Fly

- Solution is read the high byte first, then the low byte, and then the high byte again.
- If the high byte has changed, repeat the read operation

```
AGAIN: MOV A,TH1
       MOV R0,TL1
       CJNE A,TH1,AGAIN
       MOV R1,A
```

8051 Serial Communication

- The 8051 has serial communication capability built into it.
  - Full-duplex
  - Asynchronous mode only
  - Uses *data framing;* Each byte is placed in between start and stop bits



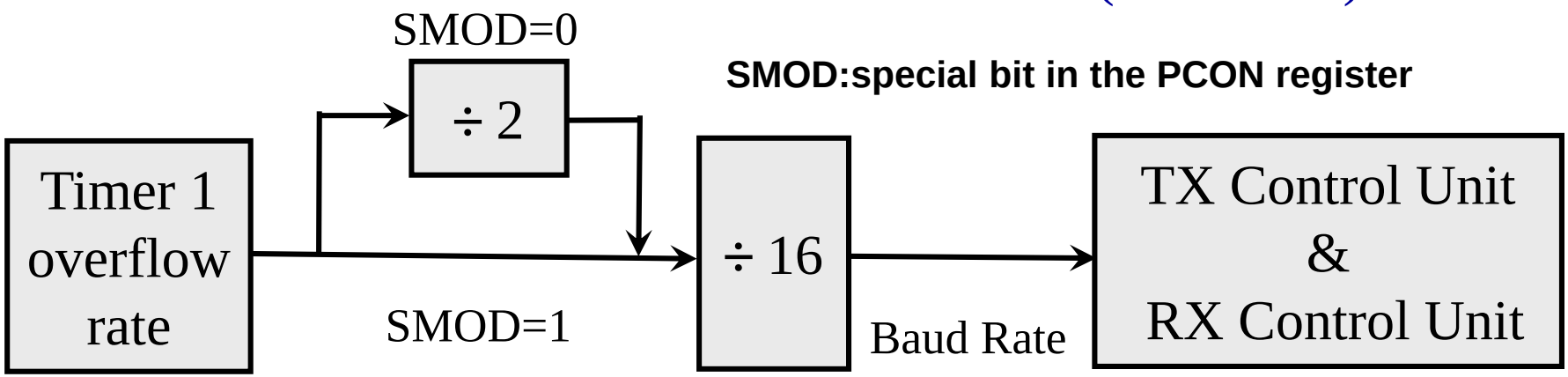
Framing

- The start bit is 0 (low) and always one bit.
- The LSB is sent out first.
- The stop bit is 1 (high).
- The stop bit can be one or two bits (one in modern PCs).
- When there is no transfer, the signal is 1 (high), which is referred to as mark.
- A total of 10 bits for each character:
  - 8-bits for the data
  - 2-bits for the start and stop bits
  - 20% overhead
- In some systems in order to maintain data integrity, the parity bit is included in the data frame.

Baud Rates

- For the 8051 to communicate with PC, the baud rates must match
  - Some PC baud rates are 110, 150, 300, 600, 1200, 2400, 4800, 9600 and 19200 (Hyperterminal supports baud rates much higher)
- The 8051 transfers and receives data serially at many different baud rates
- Baud rate is determined by Timer 1's overflow rate.(In modes 1 and 3)

$$\text{Baud Rate=Timer1 Overflow Rate}\times\frac{2^{SMOD}}{32}=\frac{\text{XTAL}\times 2^{SMOD}}{12\times(256-\text{TH1})\times 32}$$



Baud Rates in the 8051

- Timer 1, mode 2 (8-bit, auto-reload)
- Define TH1 to set the baud rate.
  - XTAL = 11.0592 MHz
  - Calculate Timer1 Overflow rate
  - For SMOD=0 Divide by 32
  - E.g., for TH1=FDH
    - Timer 1 Overflow rate: (11.0592/12)/3=307200
    - Baud rate=Timer 1 Overflow rate/32 = 9,600 Hz

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

Note: XTAL = 11.0592 MHz.

General Formula

- SMOD: special bit in the PCON register
- SMOD = 0
  - TH1 = $256 - \frac{\text{System Frequency}}{12x32x\text{Baud rate}}$
  - For example, to achieve a baud rate of 1200 using a system clock frequency of 12MHz

    TH1 = 256 - ((12MHz / (12 * 32)) / 1200)
    TH1 = 256 - 26.04
    TH1 = 256 - 26 (rounding down => error of 0.04)
    TH1 = 230
- SMOD = 1
  - TH1 = $256 - \frac{\text{System Frequency}}{12x16x\text{Baud rate}}$
- Use whichever gives less error

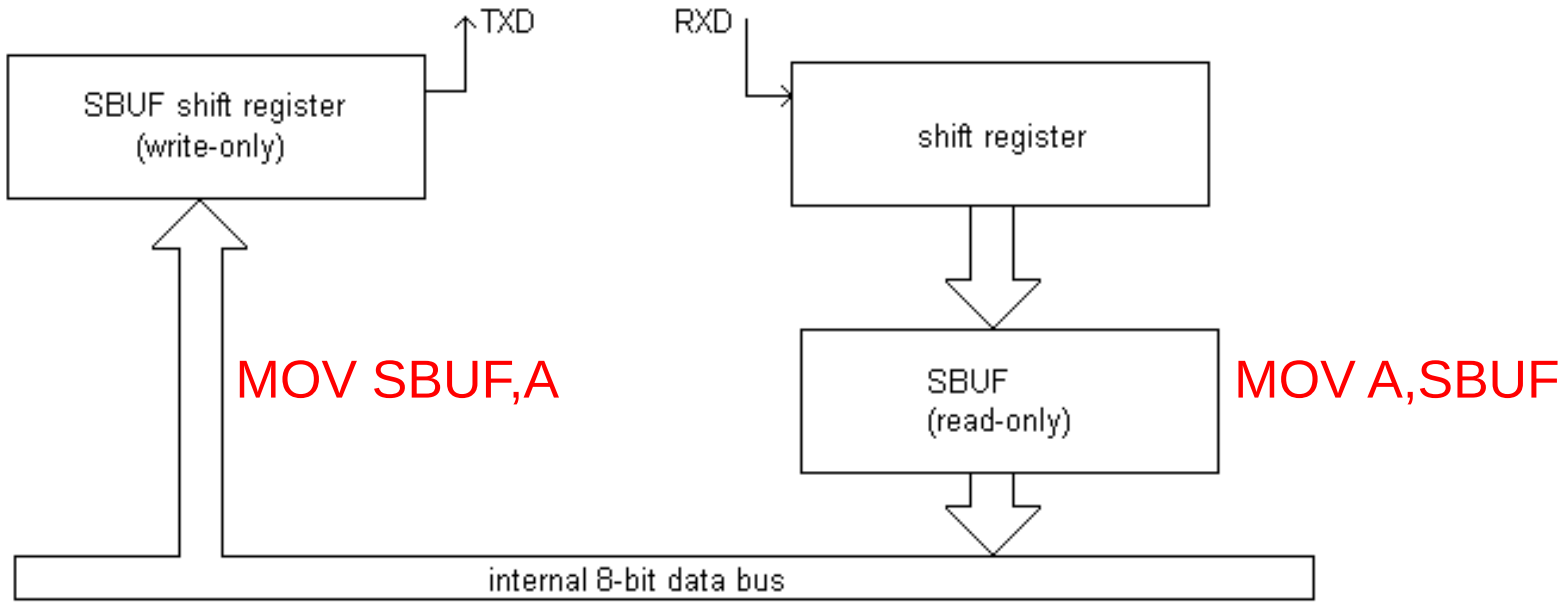
General formula

- SMOD: special bit in the PCON register
- SMOD = 0
  - $\text{TH1} = 256 - \frac{\text{System Frequency}}{12x32x\text{Baud rate}}$
  - For example, to achieve a baud rate of 1200 using a system clock frequency of 12MHz

    TH1 = 256 - ((12MHz / (12 * 32)) / 1200)
    TH1 = 256 - 26.04
    TH1 = 256 - 26 (rounding down => error of 0.04)
    TH1 = 230
- SMOD = 1
  - $\text{TH1} = 256 - \frac{\text{System Frequency}}{12x16x\text{Baud rate}}$
- Use whichever gives less error

Registers Used in Serial Comm.

- SBUF (Serial data buffer) is not bit-addressable and amounts to two distinct registers
- SCON (Serial control register) bit-addressable
- PCON (Power control register) not bit-addressable



SBUF Register

- Serial data register: **SBUF**
  - 8-bit SFR register at 99h
  - Used for transmission and reception
    - For a byte of data to be transferred via the TxD line, it must be placed in the SBUF.
    - Once it is placed, it is sent by hardware towards the TxD line at the configured baud rate
    - SBUF holds the byte of data when it is received via the 8051's RxD line.
    - SBUF is in fact two distinct registers - the write-only register and the read-only register.

```
MOV SBUF,#'D'     ;put char 'D' to transmit
MOV SBUF,A        ;send data from A
MOV A,SBUF        ;receive and copy to A
```

SCON

Bit	Name	Bit Address	Function
7	**SM0**	9Fh	Serial port mode bit 0
6	**SM1**	9Eh	Serial port mode bit 1.
5	**SM2**	9Dh	Multiprocessor Communications Enable (not needed in this course)
4	**REN**	9Ch	Receiver Enable. This bit must be set in order to receive characters.
3	**TB8**	9Bh	Transmit bit 8. The 9th bit to transmit in mode 2 and 3.
2	**RB8**	9Ah	Receive bit 8. The 9th bit received in mode 2 and 3.
1	**TI**	99h	Transmit Flag. Set when a byte has been completely transmitted.
0	**RI**	98h	Receive Flag. Set when a byte has been completely received.

SCON

- Bits 7 and 6 are used for putting the serial port into one of the four modes:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	fixed – system clock frequency divided by 12 (machine cycle frequency)
0	**1**	**1**	**8-bit UART**	**variable - set by timer 1**
1	0	2	9-bit UART	fixed - system clock frequency divided by 12 or divided by 64
1	**1**	**3**	**9-bit variable baud rate UART**	**variable - set by timer 1**

Transmission

- The 8051 lets us know when it is done transmitting a character by setting the **TI** bit in SCON.
- When this bit is set we know that the last character has been transmitted and that we may send the next character, if any.

```
BACK: CLR TI ;Be sure the bit is initially clear
MOV SBUF,@R0 ;Send the character to the serial port
JNB TI,$ ; Wait until the TI bit is set.
INC R0
SJMP BACK
```

Reception

- Reading data received by the serial port is equally easy.
- To read a byte from the serial port one just needs to read the value stored in the **SBUF** (99h) SFR after the 8051 has automatically set the **RI** flag in SCON.
- For example, if your program wants to wait for a character to be received and subsequently read it into the Accumulator, the following code segment may be used:

```
BACK: JNB RI,$ ;Wait for the 8051 to set the RI flag
MOV @R0,SBUF ;Read the character from the serial port
CLR RI
INC R0
SJMP BACK
```