8051
timer/counter

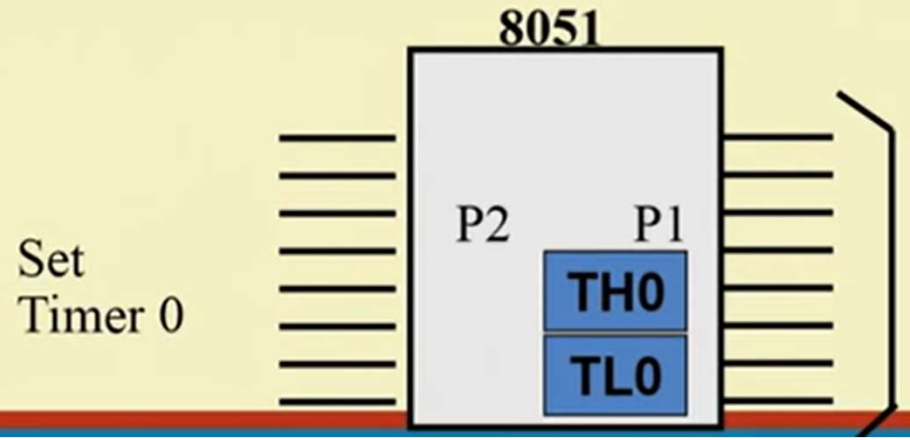
Timers /Counters Programming

- 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.

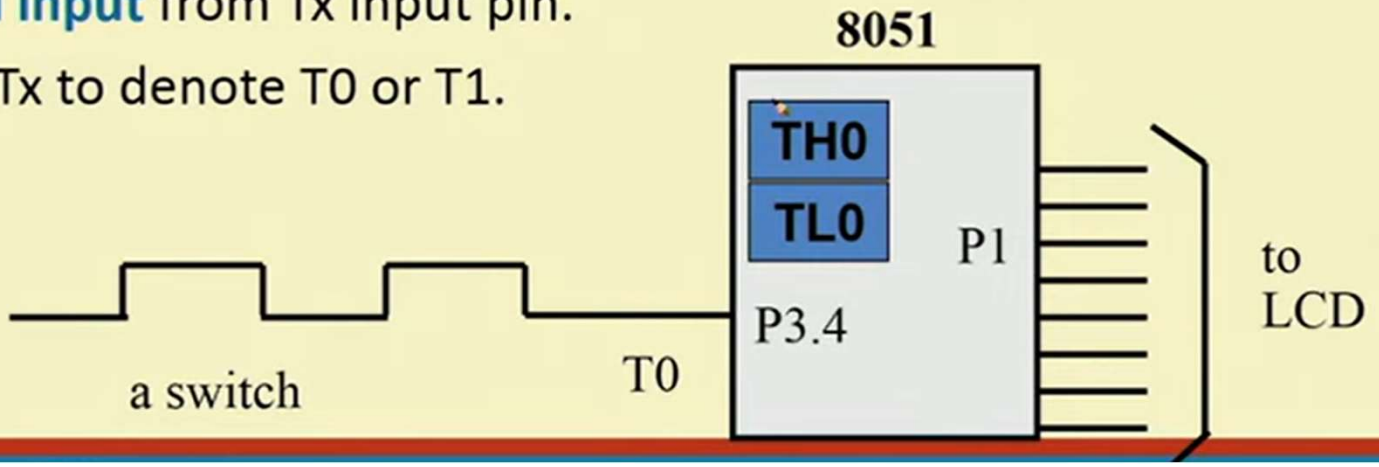
Timer

- Set the initial value of registers
- Start the timer and then the 8051 counts up.
- Input from internal system clock (machine cycle)
- When the registers equal to 0 and the 8051 sets a bit to denote time out



Counter

- Count the number of events
  - Show the number of events on registers
  - External input from T0 input pin (P3.4) for Counter 0
  - External input from T1 input pin (P3.5) for Counter 1
  - **External input** from Tx input pin.
  - We use Tx to denote T0 or T1.



Registers Used in Timer/Counter

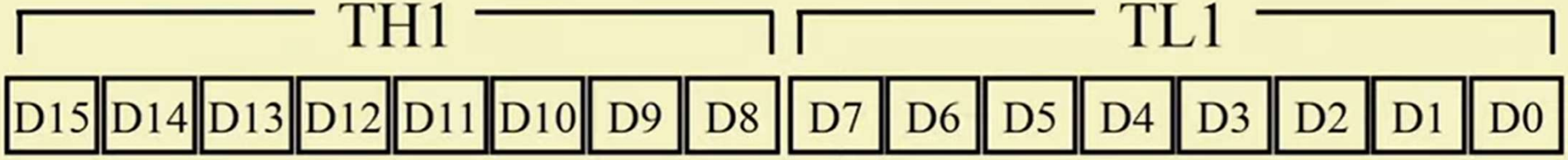
- TH0, TL0, TH1, TL1
- TMOD (Timer mode register)
- TCON (Timer control register)
- Since 8052 has 3 timers/counters, the formats of these control registers are different.
  - T2CON (Timer 2 control register), TH2 and TL2 used for 8052 only.

Basic Registers of the Timer

- Both timer 0 and timer 1 are 16 bits wide.
  - These registers stores
    - 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

Timer 0



Timer 1

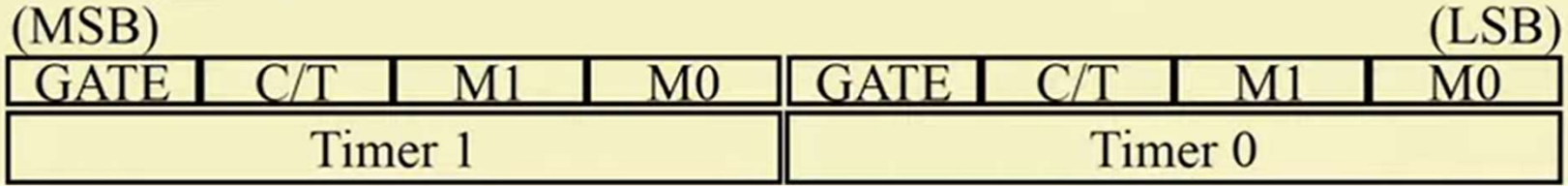
TMOD Register

GATE Gating control when set. Timer/counter is enabled only while the INTx pin is high and the TRx control pin is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T Timer or counter selected 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



Gate

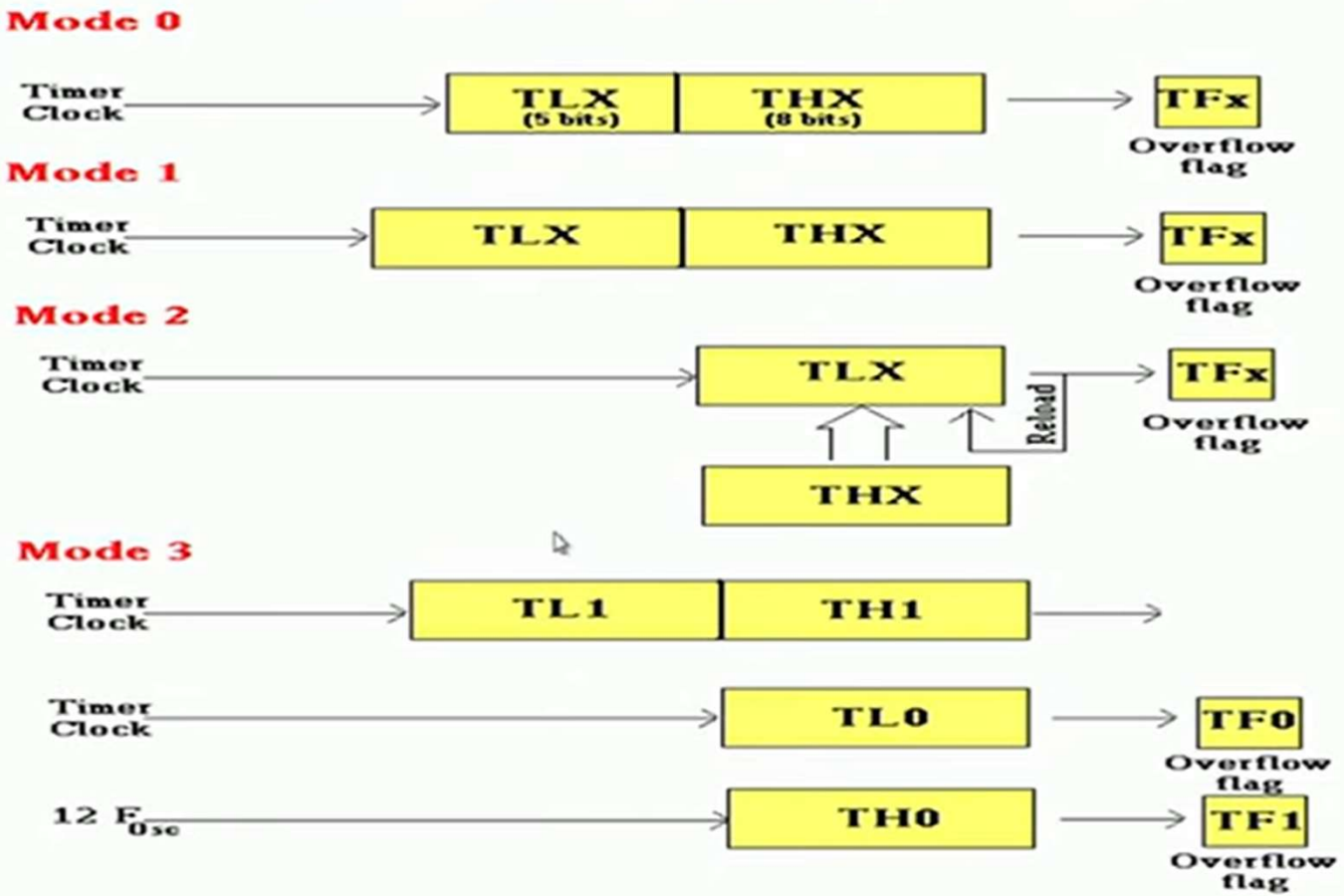
- Every timer has a mean of starting and stopping.
  - GATE=0
    - **Internal** control
    - The start and stop of the timer are controlled by way of **software.**
    - Set/clear the TR for start/stop timer.
  - GATE=1
    - **External** control
    - The hardware way of starting and stopping the 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).

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 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

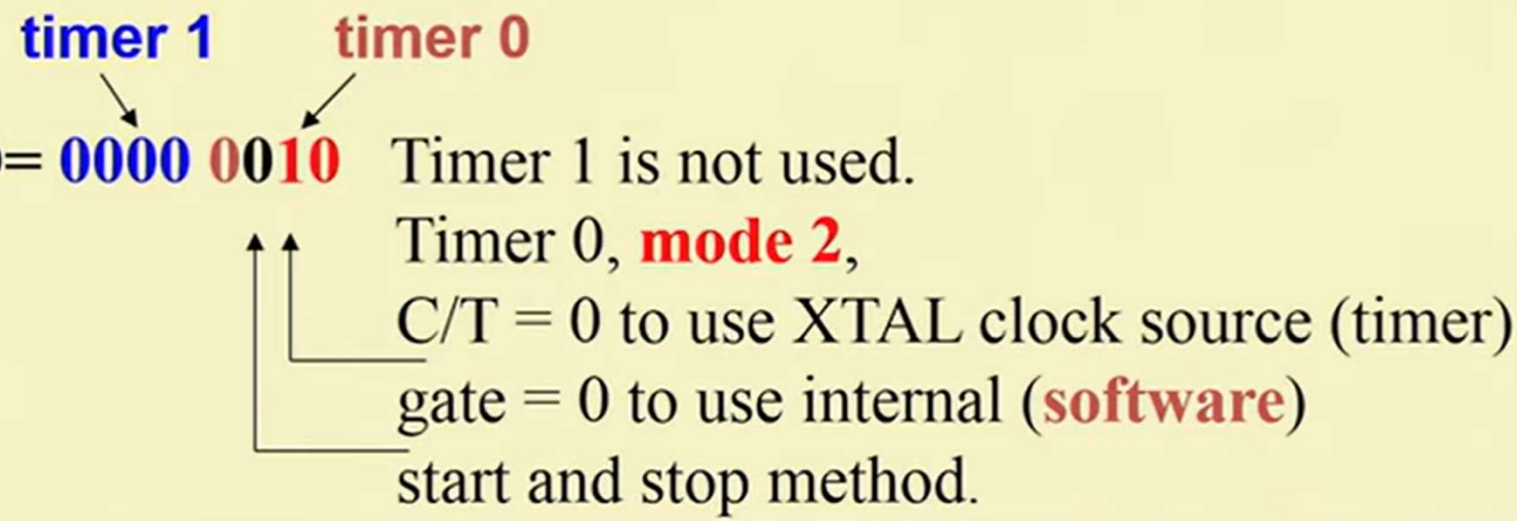
Timer modes


Mode 0
Timer Clock
TLX (5 bits)
THX (8 bits)
TFx
Overflow flag
Mode 1
Timer Clock
TLX
THX
TFx
Overflow flag
Mode 2
Timer Clock
TLX
Reload
TFx
Overflow flag
THX
Mode 3
Timer Clock
TL1
TH1
Timer Clock
TL0
TF0
Overflow flag
12 F Osc
TH0
TF1
Overflow flag

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:



TCON Register (1/2)

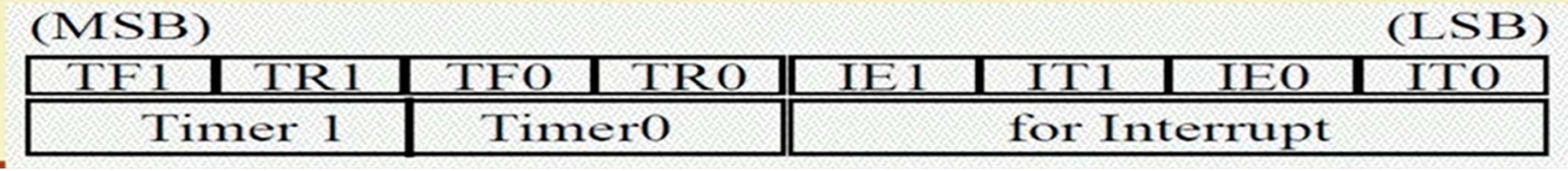
- Timer control register: **TMOD**
  - Upper nibble for timer/counter, lower nibble for interrupts
- **TR** (run control bit)
  - TR0 for Timer/counter 0; TR1 for Timer/counter 1.
  - TR is set by programmer to turn timer/counter on/off.
    - TR=0: off (stop)
    - TR=1: on (start)

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

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.



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

Timer Mode 1

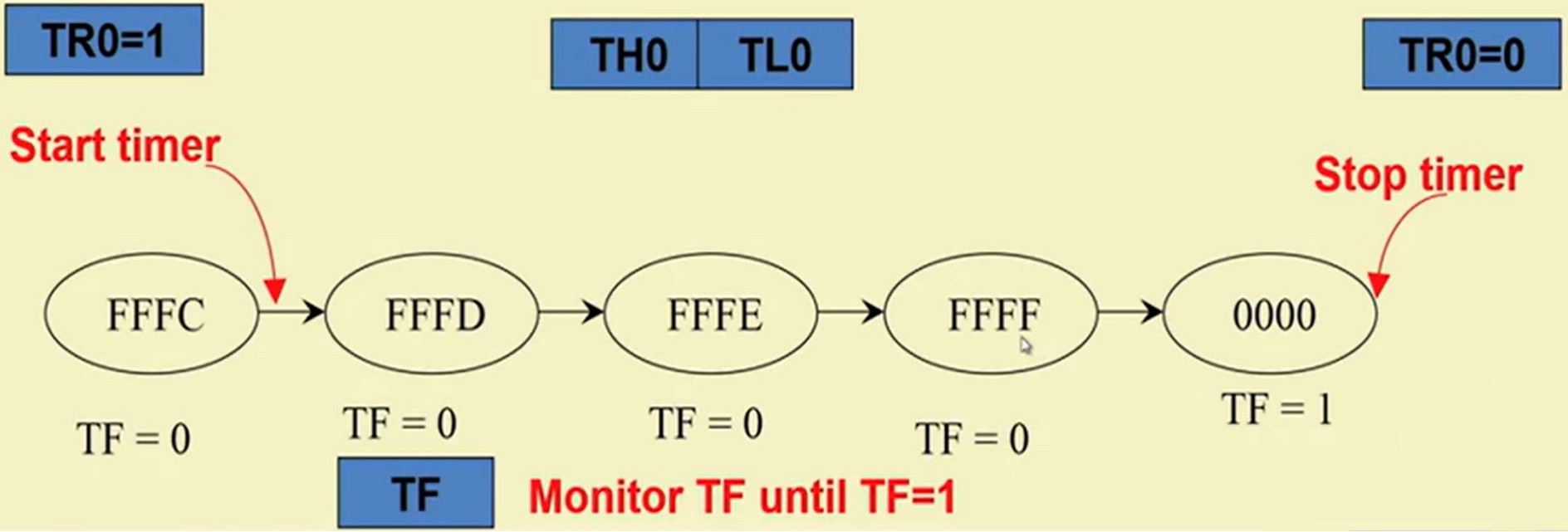
- In following, we all use timer 0 as an example.
- **16-bit** timer (TH0 and TL0)
- TH0-TL0 is incremented continuously when TR0 is set to 1. And the 8051 stops to increment TH0-TL0 when TR0 is cleared.
- The timer works with the internal system clock. In other words, the timer counts up each machine 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.

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`

Steps of Mode 1 (2/3)

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



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
```

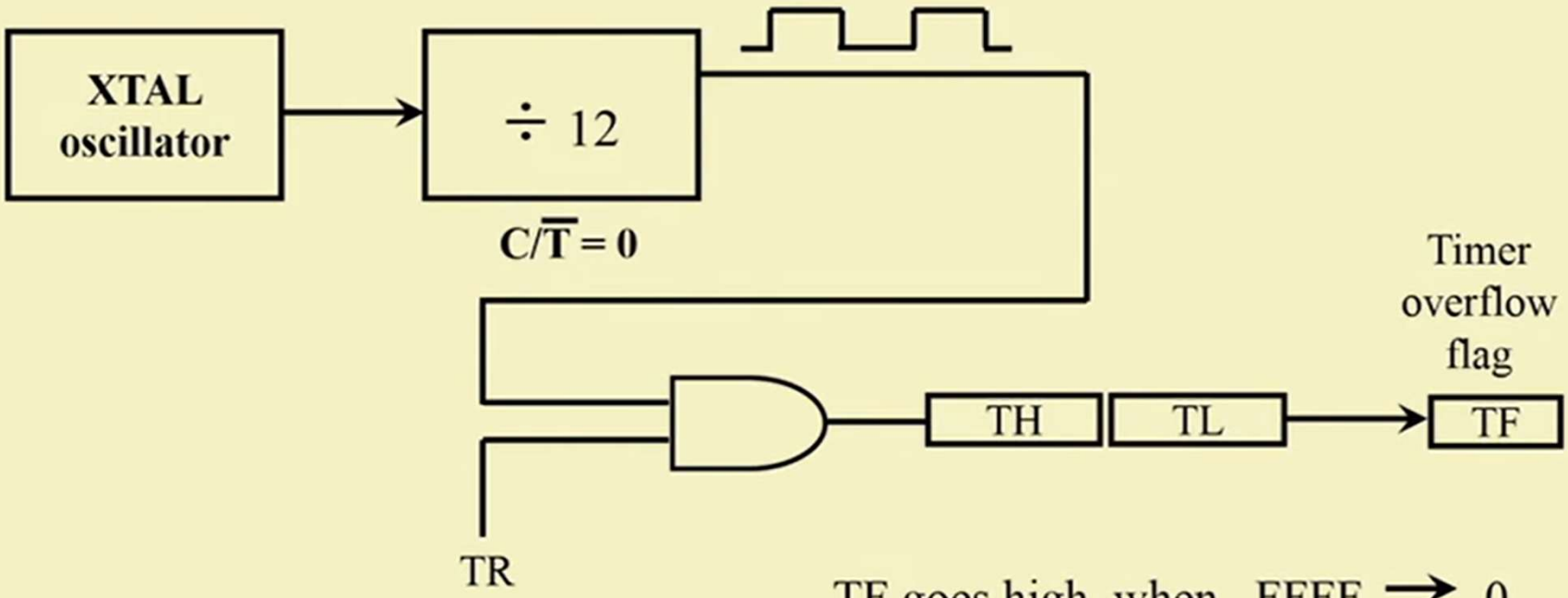
8. Clear TR0 to stop the process.

```
CLR TR0
```

9. Clear the TF flag for the next round.

```
CLR TF0
```

Mode 1 Programming



Timer Delay Calculation for XTAL = 11.0592 MHz

(a) in hex

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

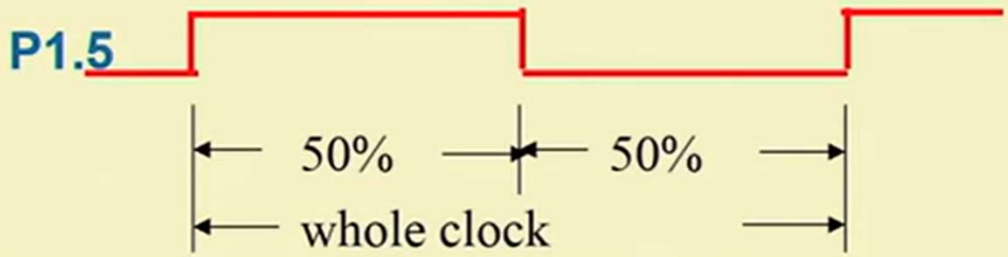
(b) in decimal

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

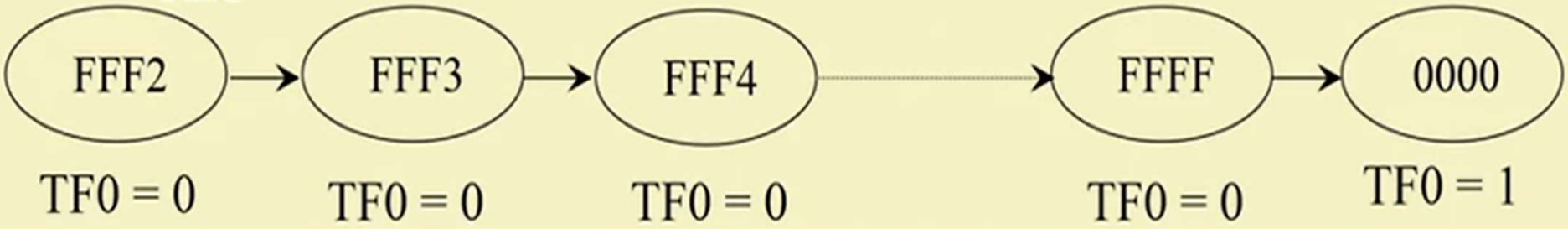
Example

- 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
```



```
;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
```



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
```

Example

Solution:

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

Half period = 35276 × 1.085 μs = 38.274 ms

Whole period = 2 × 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.

Find Timer Values

- Assume that XTAL = 11.0592 MHz .
- And we know desired **delay**
- how to find the values for the TH,TL ?
    1. Divide the **delay** by **1.085 μs** and get **n.**
    2. Perform **65536 –n**
    3. Convert the result of Step 2 to hex (**yyxx**)
    4. Set **TH = yy** and **TL = xx**.

Example

- Assuming XTAL = 11.0592 MHz,
- write a program to generate a square wave of 50 Hz frequency on pin P2.3.

Solution:

1. The period of the square wave = **1 / 50 Hz = 20 ms.**
2. The high or low portion of the square wave = **10 ms**.
3. **10 ms / 1.085 μs = 9216**
4. **65536 – 9216 = 56320** in decimal = **DC00H** in hex.
5. TL1 = 00H and TH1 = DCH.

Example

```
        MOV   TMOD,#10H     ;timer 1, mode 1
AGAIN:  MOV   TL1,#00        ;Timer value = DC00H
        MOV   TH1,#0DCH
        SETB  TR1            ;start
BACK:   JNB   TF1,BACK
        CLR   TR1            ;stop
        CPL   P2.3
        CLR   TF1            ;clear timer flag 1
        SJMP  AGAIN          ;reload timer since
                             ;mode 1 is not
                             ;auto-reload
```

Timer Mode 0

- Mode 0 is exactly like mode 1 except that it is a **13-bit** timer instead of 16-bit.
  - 8-bit TH0
  - 5-bit TL0
- The counter can hold values between 0000 to 1FFF in TH0-TL0.
  - 2^{13}-1= 2000H-1=1FFFH
- We set the initial values TH0-TL0 to count up.
- When the timer reaches its maximum of 1FFFH, it rolls over to 0000, and TF0 is raised.

Timer Mode 2

- 8-bit timer.
  - It allows only values of 00 to FFH to be loaded into TH0.
- Auto-reloading
- TL0 is incremented continuously when TR0=1.

Steps of Mode 2

1. Chose mode 2 timer 0

```
MOV TMOD,#02H
```

2. Set the original value to TH0.

```
MOV TH0,#38H
```

3. Clear the flag to TF0=0.

```
CLR TF0
```

4. After TH0 is loaded with the 8-bit value, the 8051 gives a copy of it to TL0.

```
TL0=TH0=38H
```

5. Start the timer.

```
SETB TR0
```

Steps of Mode 2

6. The 8051 starts to count up by incrementing the TL0.
   - `TL0= 38H, 39H, 3AH,....`
7. When TL0 rolls over from FFH to 00, the 8051 set TF0=1. Also, TL0 is reloaded automatically with the value kept by the TH0.
   - `TL0= FEH, FFH, 00H (Now TF0=1)`
   - The 8051 auto reload `TL0=TH0=38H.`
   - `Clr TF0`
   - 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.
  - `Clr TR0`

Counter

- These timers can also be used as counters **counting events** happening outside the 8051.
- When the timer is used as a counter, it is a pulse outside of the 8051 that increments the TH, TL.
- When C/T=1, the counter counts up as pulses are fed from
  - T0: timer 0 input (Pin 14, P3.4)
  - T1: timer 1 input (Pin 15, P3.5)

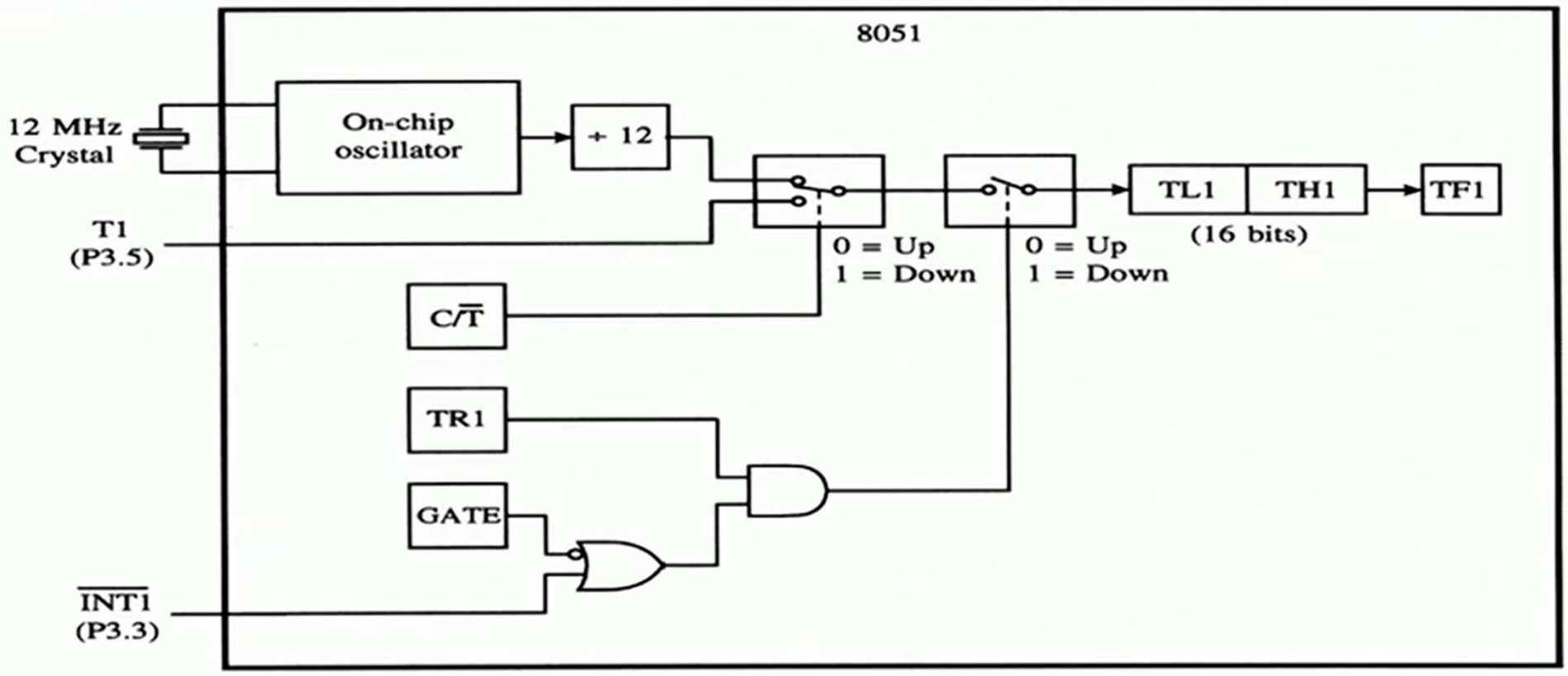
Port 3 Pins Used For Timers 0 and 1

Pin	Port Pin	Function	Description
14	P3.4	T0	Timer/Counter 0 external input
15	P3.5	T1	Timer/Counter 1 external input

(MSB) (LSB)

GATE	C/T=1	M1	M0	GATE	C/T=1	M1	M0
Timer 1				Timer 0			

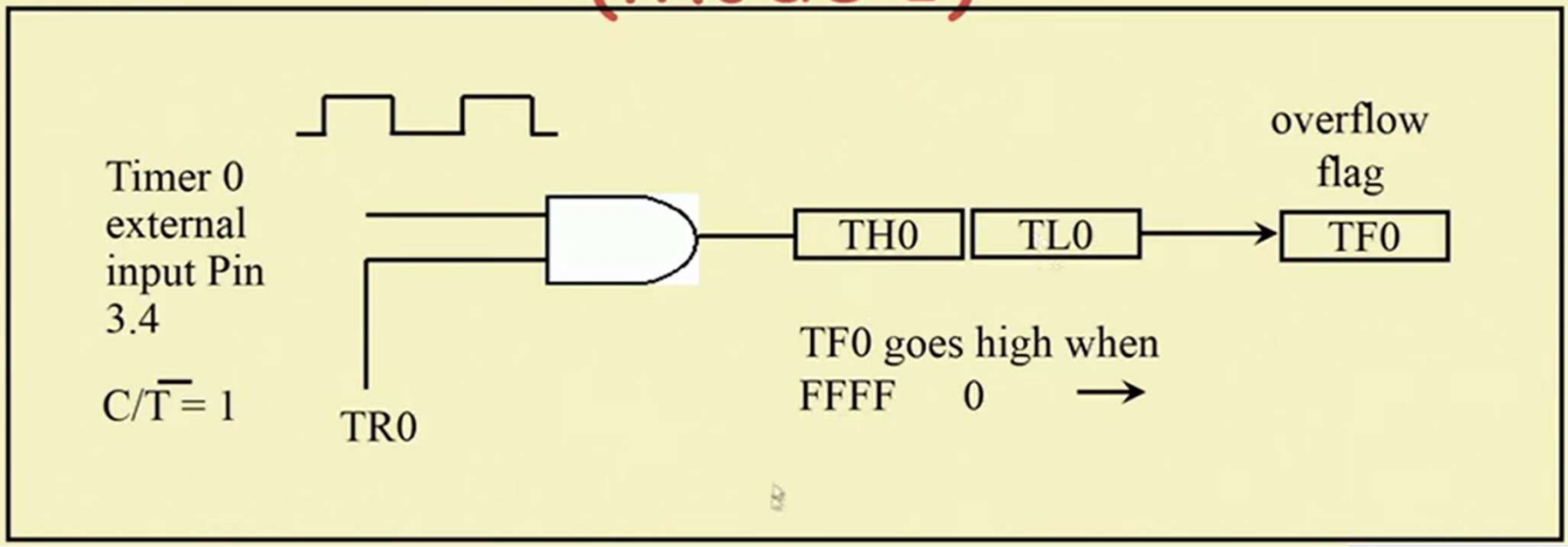
Timer/Counter selection



Counter Mode 1

- **16-bit** counter (TH0 and TL0)
- TH0-TL0 is incremented when TR0 is set to 1 **and** an external pulse (in 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=1 as an indicator to show a special condition. (ex: 100 people have come).

Timer 0 with External Input (Mode 1)



Counter Mode 2

- 8-bit **counter**.
  - It allows only values of 00 to FFH to be loaded into TH0.
- Auto-reloading
- TL0 is incremented if TR0=1 and external pulse occurs.

Example

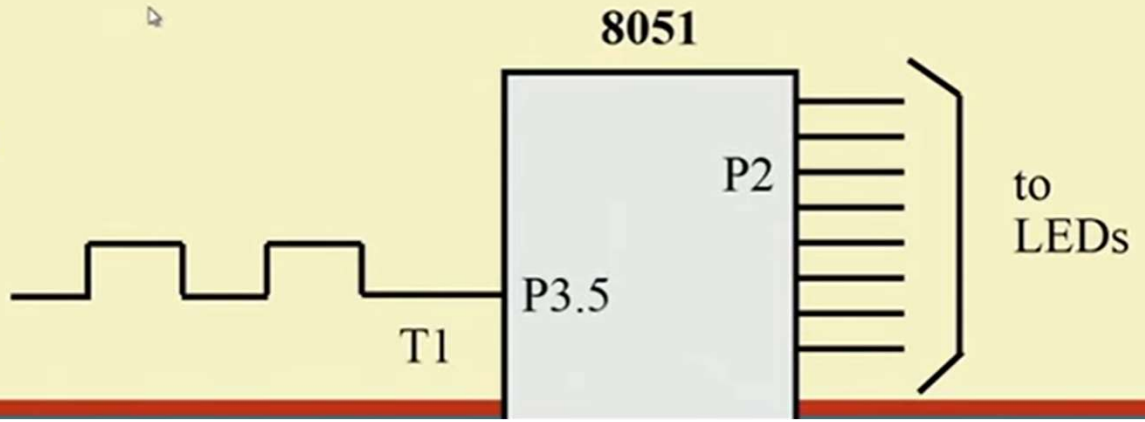
Assuming that clock pulses are fed into pin T1, write a program for counter 1 in mode 2 to count the pulses and display the state of the TL 1 count on P2.

```
      MOV  TMOD,#01100000B    ;mode 2, counter 1
      MOV  TH1,#0
      SETB P3.5               ;make T1 input port
AGAIN:SETB TR1                ;start
BACK: MOV  A,TL1
      MOV  P2,A               ;display in P2
      JNB  TF1,Back           ;overflow
      CLR  TR1                ;stop
      CLR  TF1                ;make TF=0
      SJMP AGAIN              ;keep doing it
```

Example

- Timer 1 as an event counter fed into pin3.5.
- “SETB P3.5” make P3.5 an input port by making it high

P2 is connected to 8 LEDs
and input T1 to pulse.



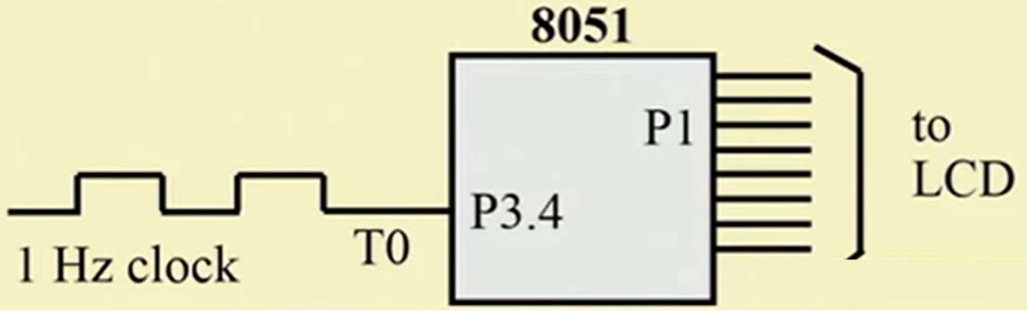
Example

Assume that a **1-Hz frequency pulse** is connected to input pin 3.4.

Write a program to display counter 0 on an LCD. Set the initial value of TH0 to -60.

Solution:

Note that on the first round, it starts from 0 and counts 256 events, since on RESET, TL0=0. To solve this problem, load TH0 with -60 at the beginning of the program.



Example

```
        ACALL LCD_SET_UP        ;initialize the LCD
        MOV   TMOD,#00000110B   ;Counter 0,mode2
        MOV   TH0,#-60
        SETB  P3.4              ;make T0 as input
AGAIN:  SETB  TR0               ;starts the counter
BACK:   MOV   A,TL0             ;every 60 events
        ACALL CONV              ;convert in R2,R3,R4
        ACALL DISPLY            ;display on LCD
        JNB   TF0,BACK          ;loop if TF0=0
        CLR   TR0               ;stop
        CLR   TF0
        SJMP  AGAIN
```

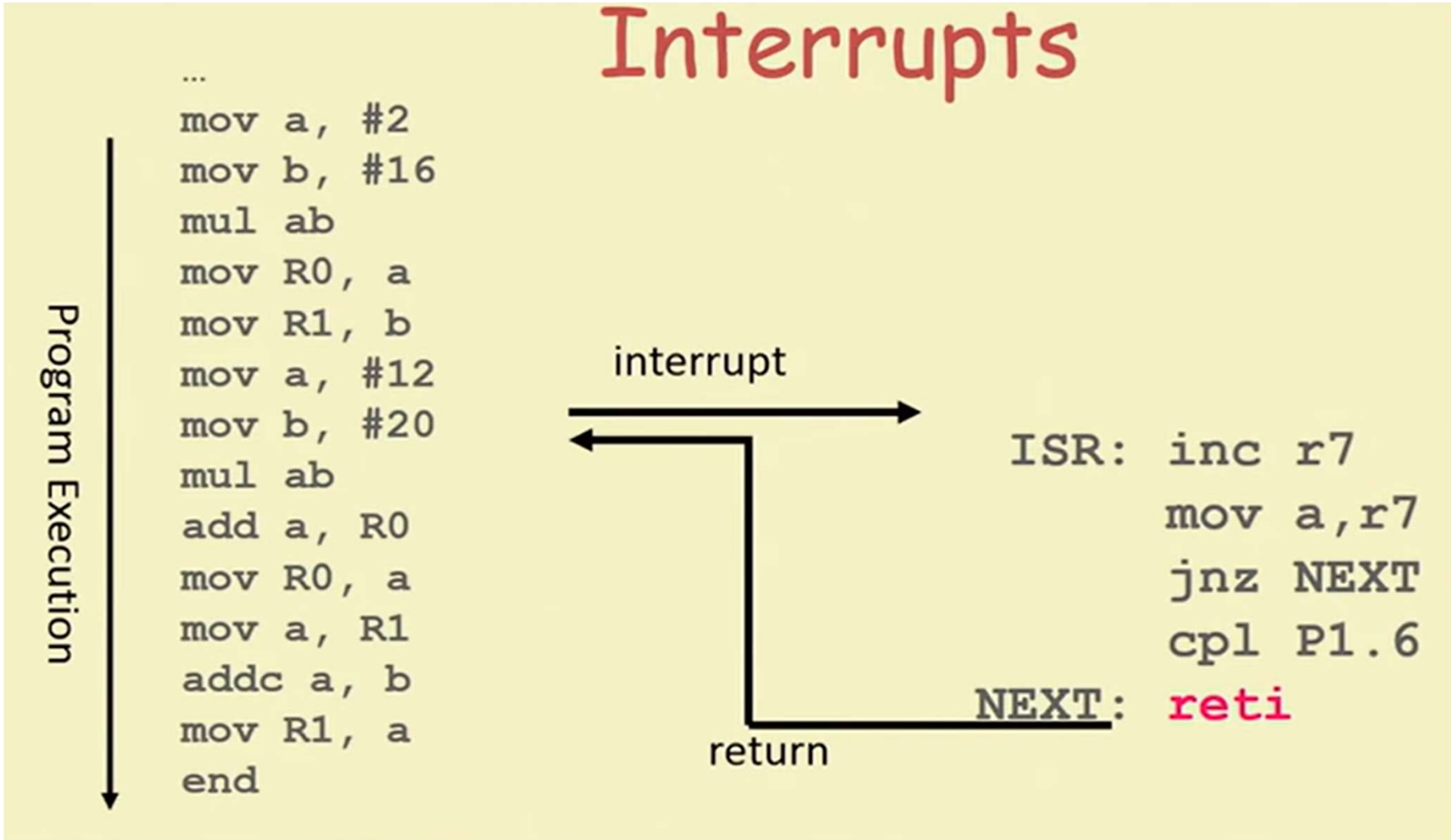
GATE=1 in TMOD

- All discuss so far has assumed that GATE=0.
  - The timer is stared with instructions "**`SETB TR0`**" and "**`SETB TR1`**" for timers 0 and 1, respectively.
- If GATE=1, we can use hardware to control the start and stop of the timers.
  - INT0 (P3.2, pin 12) starts and stops timer 0
  - INT1 (P3.3, pin 13) starts and stops timer 1
  - This allows us to start or stop the timer externally at any time via a simple switch.

GATE (external control)

- Timer 0 must be turned on by "`SETB TR0`"
- If GATE=1 count up if
  - INT0 input is high
  - TR0=1
- If GATE=0 count up if
  - TR0=1

Interrupts


...
mov a, #2
mov b, #16
mul ab
mov R0, a
mov R1, b
mov a, #12
mov b, #20
mul ab
add a, R0
mov R0, a
mov a, R1
addc a, b
mov R1, a
end
Program Execution
interrupt
return
ISR: inc r7
mov a,r7
jnz NEXT
cpl P1.6
NEXT: reti

Interrupt Sources

- Original 8051 has 5 sources of interrupts
  - Timer 0 overflow
  - Timer 1 overflow
  - External Interrupt 0
  - External Interrupt 1
  - Serial Port events (buffer full, buffer empty, etc)
- Enhanced version has 22 sources
  - More timers, programmable counter array, ADC, more external interrupts, another serial port (UART)

Interrupt Process

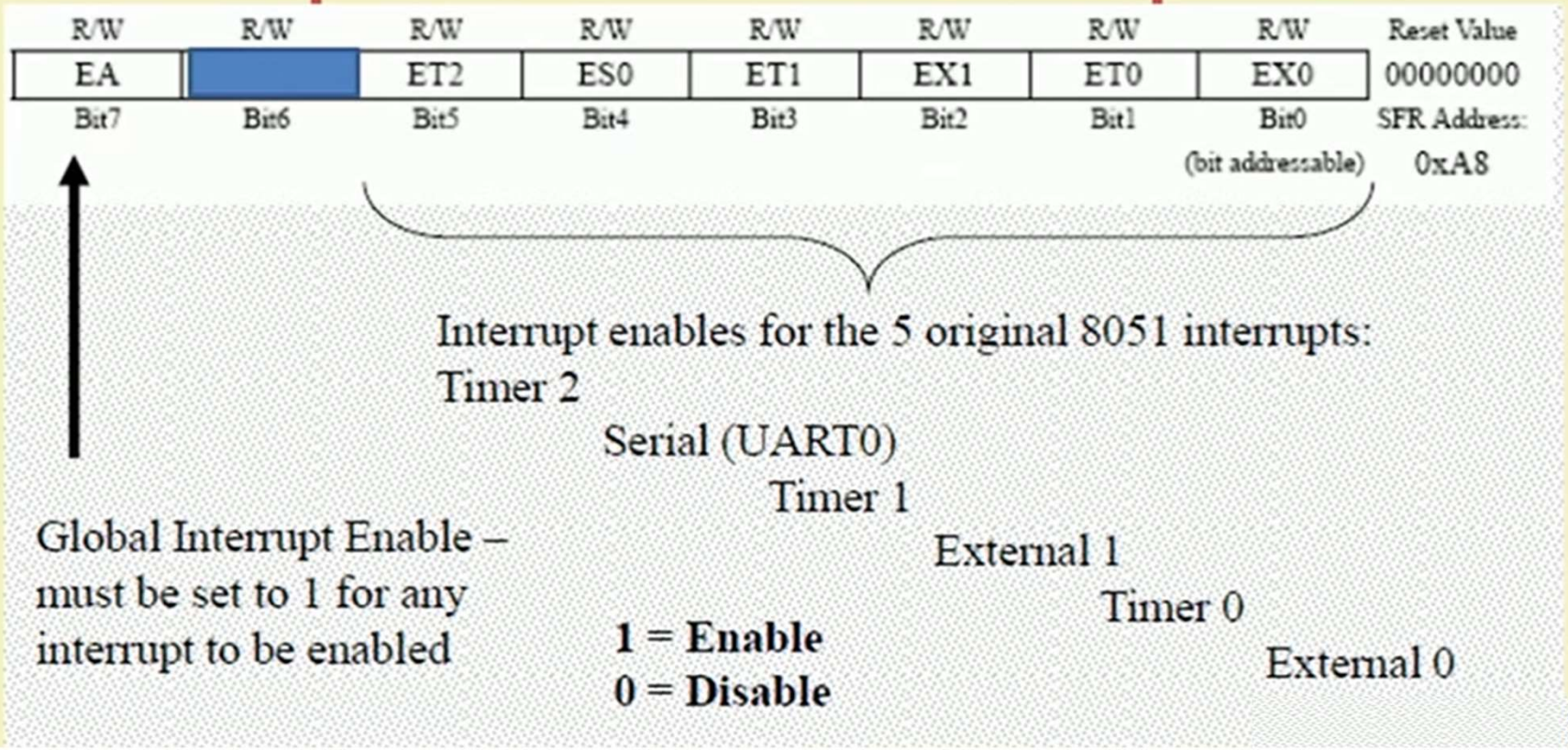
If interrupt event occurs AND interrupt flag for that event is enabled, AND interrupts are enabled, then:

1. Current PC is pushed on stack.
2. Program execution continues at the interrupt vector address for that interrupt.
3. When a RETI instruction is encountered, the PC is popped from the stack and program execution resumes where it left off.

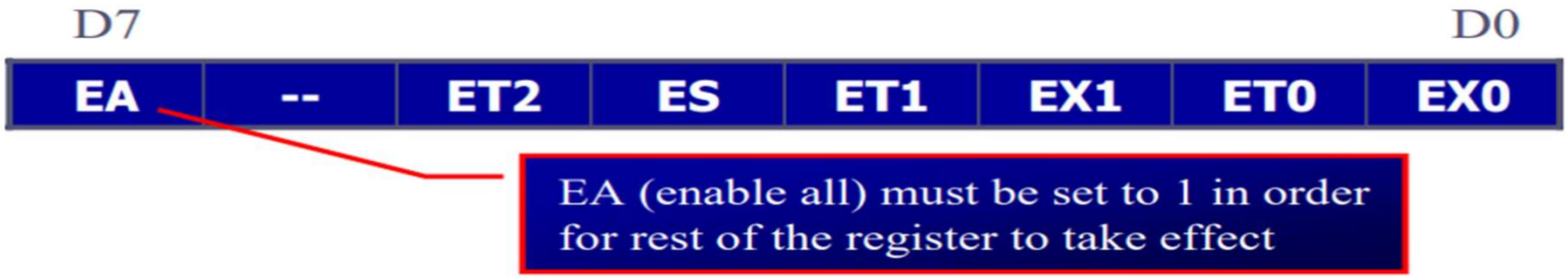
Interrupt Priorities

- What if two interrupt sources interrupt at the same time?
- The interrupt with the highest PRIORITY gets serviced first.
- All interrupts have a default priority order.
- Priority can also be set to “high” or “low”

Interrupt SFR - Interrupt Enable

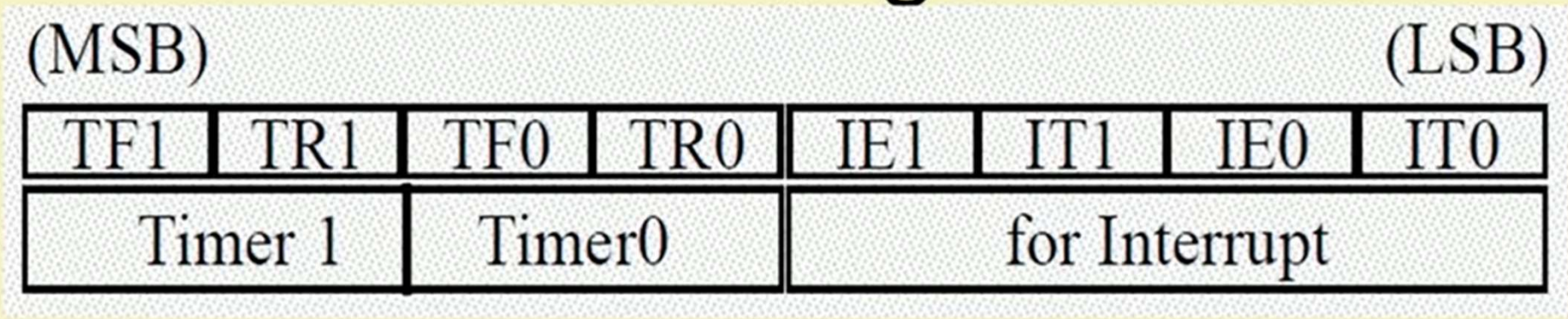


IE (Interrupt Enable) Register



EA	**IE.7**	**Disables all interrupts**
--	**IE.6**	**Not implemented, reserved for future use**
ET2	**IE.5**	**Enables or disables timer 2 overflow or capture interrupt (8952)**
ES	**IE.4**	**Enables or disables the serial port interrupt**
ET1	**IE.3**	**Enables or disables timer 1 overflow interrupt**
EX1	**IE.2**	**Enables or disables external interrupt 1**
ET0	**IE.1**	**Enables or disables timer 0 overflow interrupt**
EX0	**IE.0**	**Enables or disables external interrupt 0**

TCON Register



IE1: Set by CPU when H-L transition detected on external interrupt 1. Cleared when processed.

IT1: Interrupt 1 type control. Set/cleared by software to specify falling edge/low-level triggered.

Similarly IE0 and IT0.

Interrupt Priorities

- Default priority: INT0 > TF0 > INT1 > TF1 > RI + TI
- To alter priority set IP register

--	--	PT2	PS	PT1	PX1	PT0	PX0

- PT2: Priority of timer 2 interrupt
- PS: Priority for serial port interrupt, and so on.
- “MOV IP, #00000100B” sets INT1 to have the highest priority
- “MOV IP, #00001100B” sets priorities as INT1 > TF1 > INT0 > TF0 > RI + TI

Interrupt Priorities

- Default priority: INT0 > TF0 > INT1 > TF1 > RI + TI
- To alter priority set IP register

--	--	PT2	PS	PT1	PX1	PT0	PX0

- PT2: Priority of timer 2 interrupt
- PS: Priority for serial port interrupt, and so on.
- “MOV IP, #00000100B” sets INT1 to have the highest priority
- “MOV IP, #00001100B” sets priorities as INT1 > TF1 > INT0 > TF0 > RI + TI

Interrupt Priority Register (Bit-addressable)

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

Priority bit=1 assigns high priority
Priority bit=0 assigns low priority

Interrupt Vectors

Each interrupt has a specific place in code memory where program execution (interrupt service routine) begins.

Reset:	**0000h**
External Interrupt 0:	**0003h**
Timer 0 overflow:	**000Bh**
External Interrupt 1:	**0013h**
Timer 1 overflow:	**001Bh**
Serial :	**0023h**
Timer 2 overflow(8052+)	**002bh**

Note: that there are only 8 memory locations between vectors.

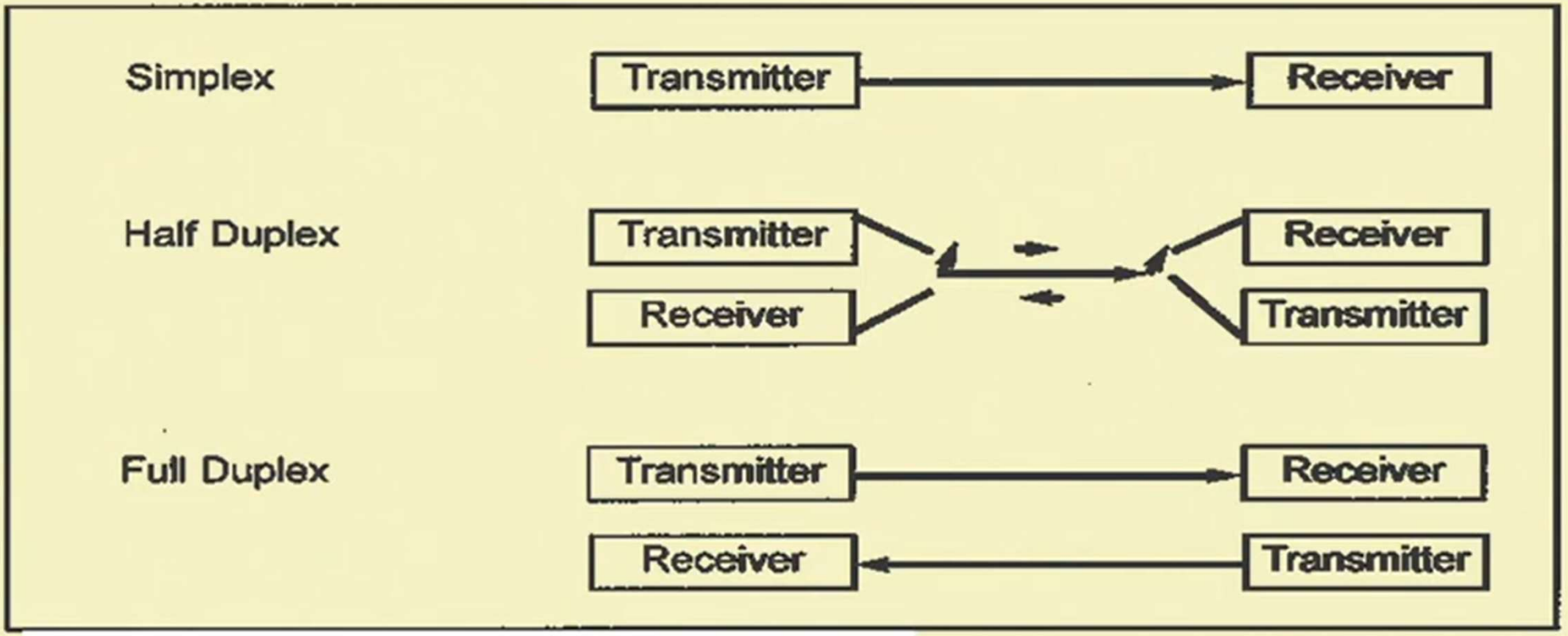
Example Interrupt Service Routine

Pin 3.3 (INT1) is connected to a pulse generator. Write a program in which the falling edge of the pulse will send a high to P1.3, connected to a LED.

```
        org 0000h
        ljmp main
; ISR for hardware interrupt INT1
        org 0013h
        setb p1.3
        mov r3, #255
back:   djnz r3, back
        clr P1.3
        reti
; Main program for initialization
        org 30h
main:   setb tcon.2       ; make INT1 edge triggered
        mov ie, #10000100B        ; enable int1
here:   sjmp here
```

Serial Communication

Basics of serial communication



- **A *protocol* is a set of rules agreed by both the sender and receiver on**
  - **How the data is packed**
  - **How many bits constitute a character**
  - **When the data begins and ends**
- **Asynchronous serial data communication is widely used for character-oriented transmissions**
  - **Each character is placed in between start and stop bits, this is called *framing***
  - **Block-oriented data transfers use the synchronous method**
- **The start bit is always one bit, but the stop bit can be one or two bits**

Start and stop bits

- When there is no transfer the signal is high

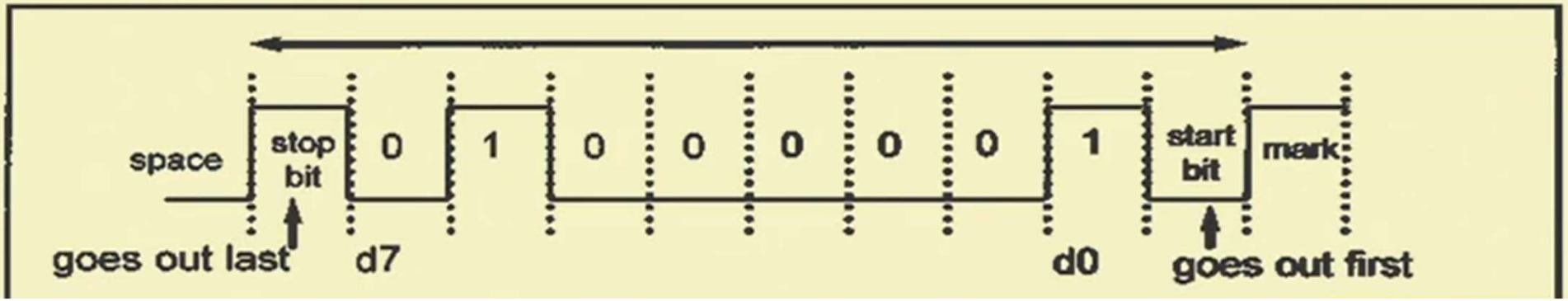
Transmission begins with a start (low) bit

LSB first

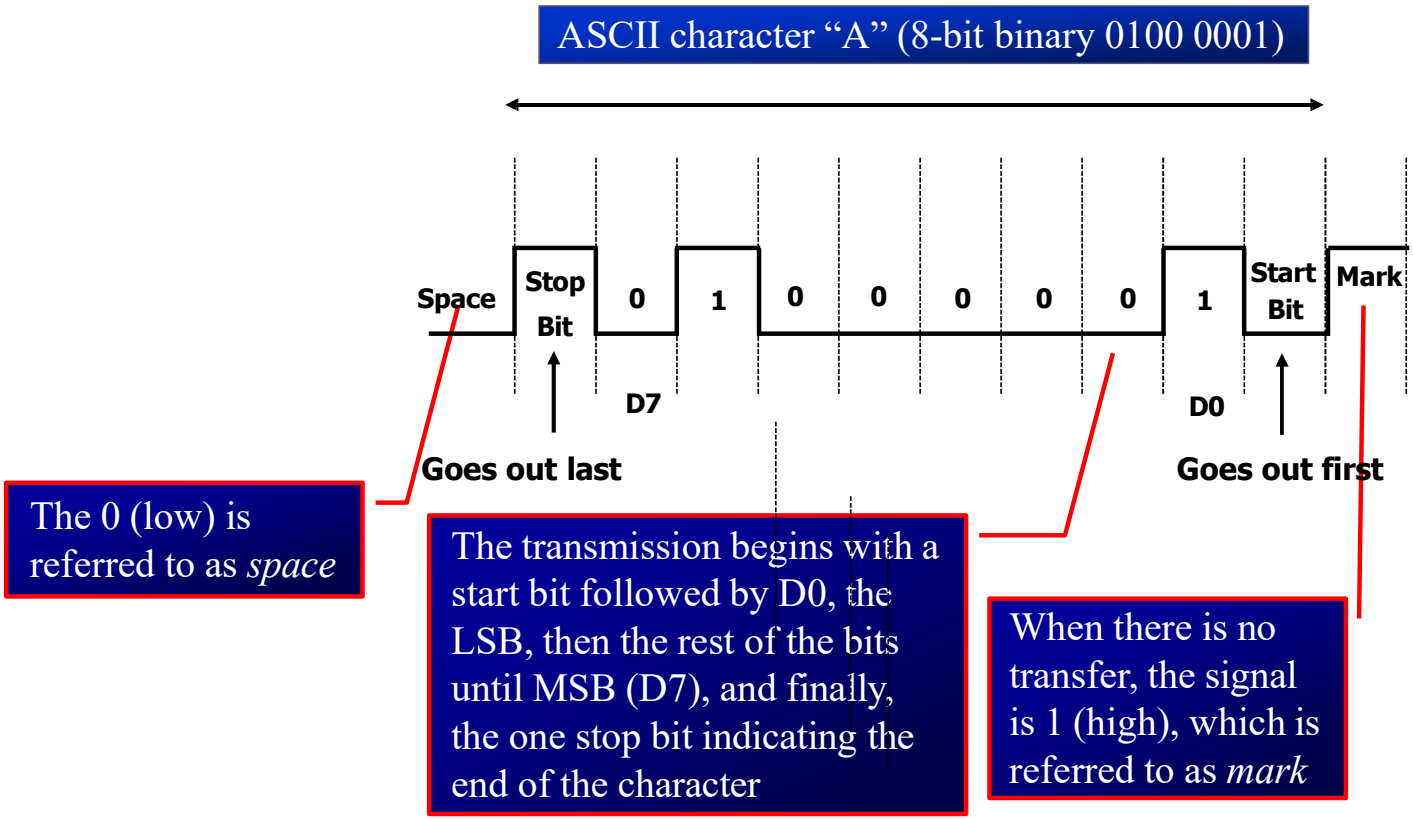
Finally 1 stop bit (high)

Data transfer rate (baud rate) is stated in bps

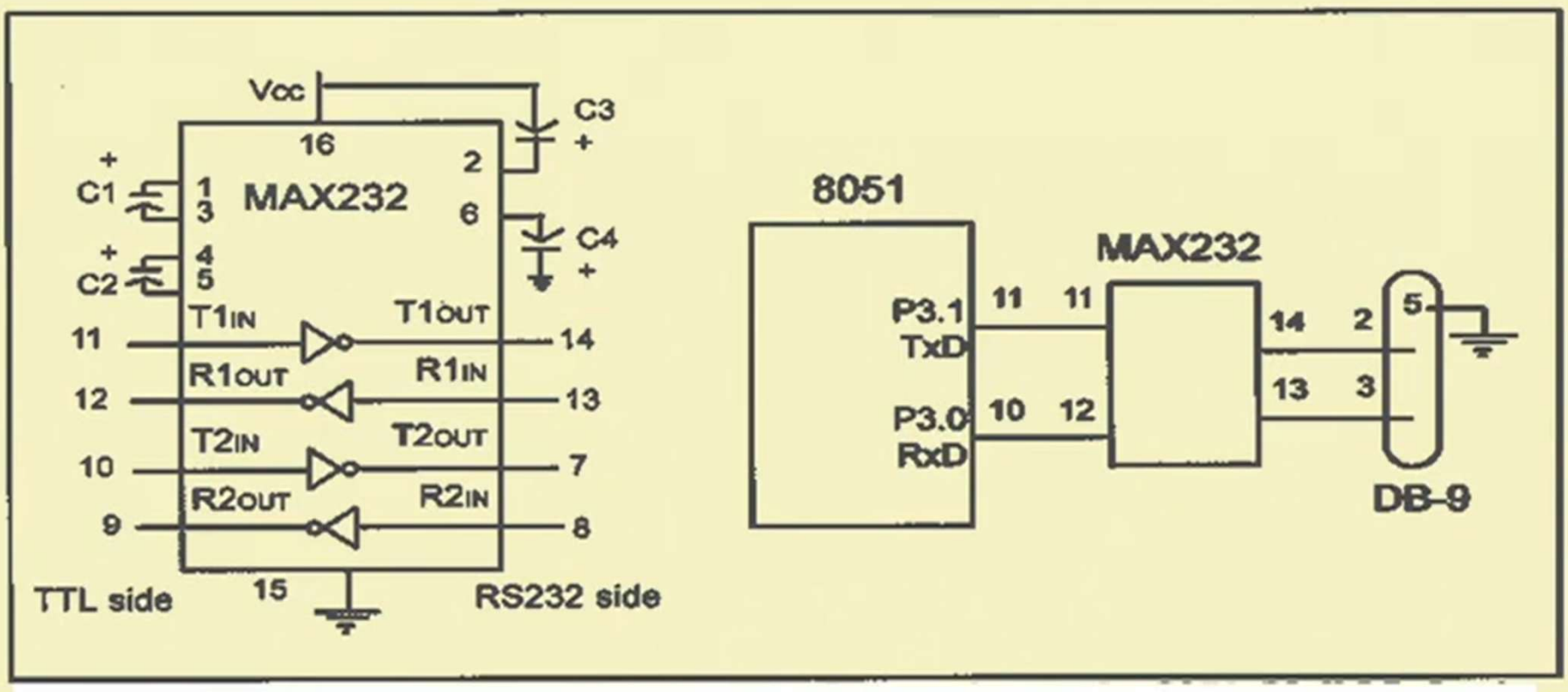
bps: bit per second



The start bit is always a 0 (low) and the stop bit(s) is 1 (high)



MAX232


Vcc
16
C3
+
2
C1
+
1
3
MAX232
6
C4
+
C2
+
4
5
T1IN
T1OUT
11
14
R1OUT
R1IN
12
13
T2IN
T2OUT
10
7
R2OUT
R2IN
9
8
15
TTL side
RS232 side
8051
MAX232
P3.1
TxD
P3.0
RxD
11
11
10
12
14
2
13
3
5
DB-9

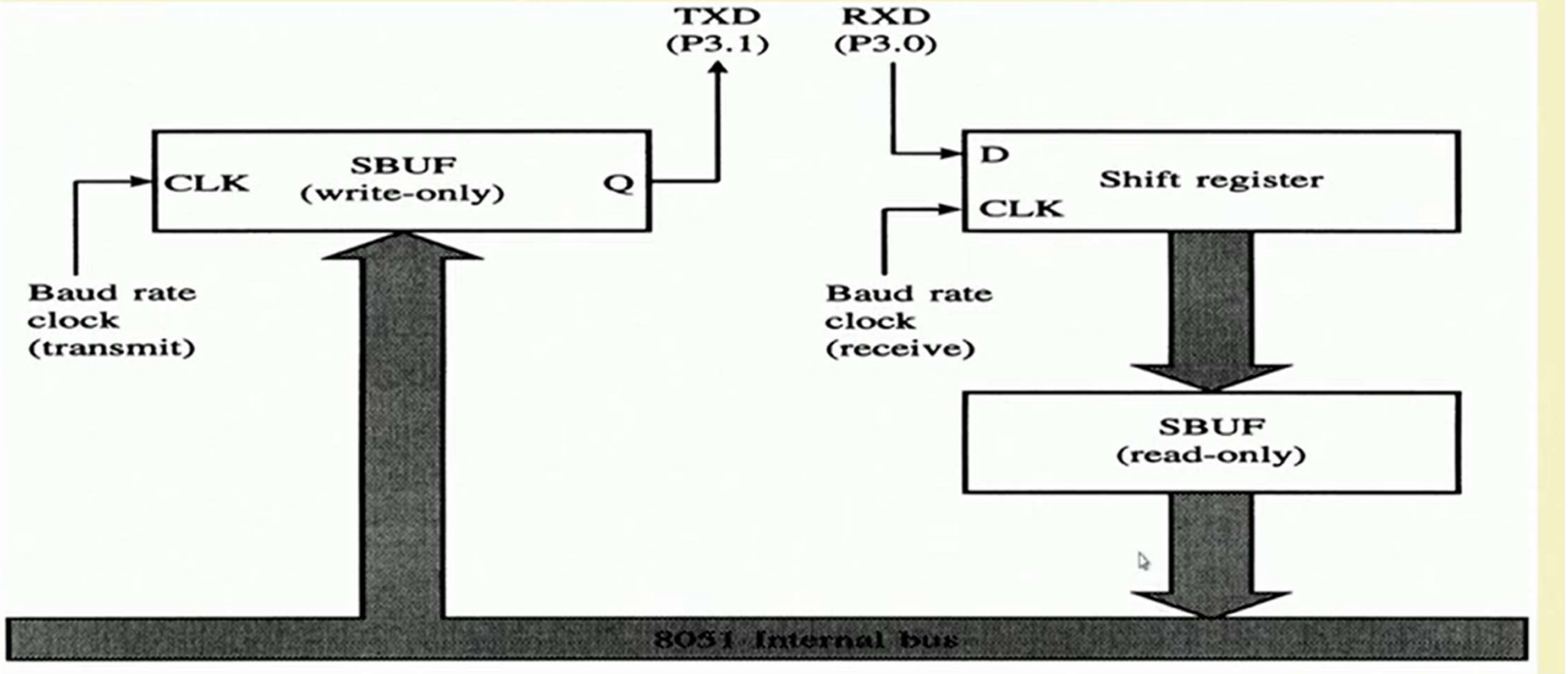
RxD and TxD pins in the 8051

- TxD pin 11 of the 8051 (P3.1)
- RxD pin 10 of the 8051 (P3.0)

SBUF register

```
MOV     SBUF,#'D'      ;load SBUF=44H, ASCII for 'D'
MOV     SBUF,A          ;copy accumulator into SBUF
MOV     A,SBUF          ;copy SBUF into accumulator
```

Serial port block diagram



- **To allow data transfer between the PC and an 8051 system without any error, we must make sure that the baud rate of 8051 system matches the baud rate of the PC ś COM port**
- **Hyperterminal function supports baud rates much higher than listed below**

PC Baud Rates
110
150
300
600
1200
2400
4800
9600
19200

Baud rates supported by 486/Pentium IBM PC BIOS

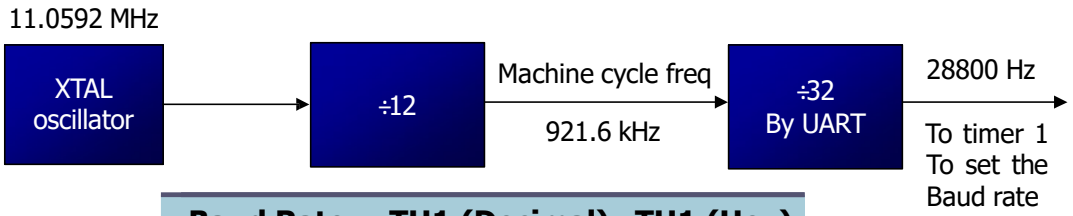
With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates. (a) 9600 (b) 2400 (c) 1200

Solution:

The machine cycle frequency of 8051 = 11.0592 / 12 = 921.6 kHz, and 921.6 kHz / 32 = 28,800 Hz is frequency by UART to timer 1 to set baud rate.

(a) 28,800 / 3 = 9600 where -3 = FD (hex) is loaded into TH1
(b) 28,800 / 12 = 2400 where -12 = F4 (hex) is loaded into TH1
(c) 28,800 / 24 = 1200 where -24 = E8 (hex) is loaded into TH1

Notice that dividing 1/12 of the crystal frequency by 32 is the default value upon activation of the 8051 RESET pin.



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

TF is set to 1 every 12 ticks, so it functions as a frequency divider

Serial control (SCON) Register

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 (SCON.7) : mode specifier

SM1 (SCON.6) : mode specifier

SM2 (SCON.5) : used for multi processor communication

REN (SCON.4) : receive enable (by software enable/disable)

TB8 (SCON.3) : transmit bit8

RB8 (SCON.2) : receive bit 8

TI (SCON.1) : transmit interrupt flag set by HW clear by SW

RI (SCON.0) : receive interrupt flag set by HW clear by SW

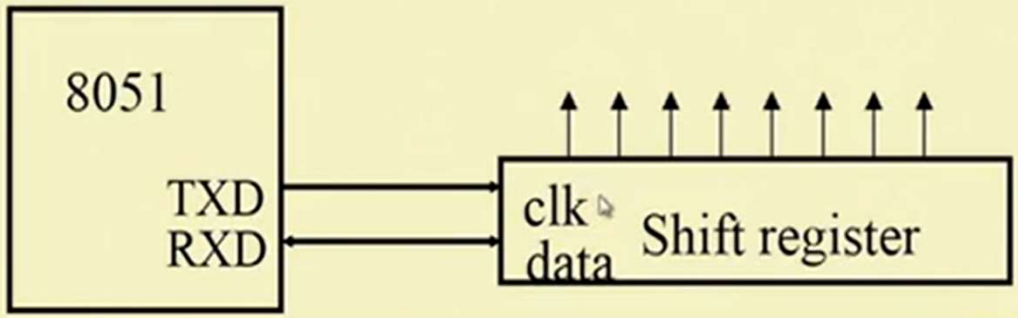
Mode of operation

SM0	SM1	MODE	operation	transmit rate
0	0	0	shift register	fixed (xtal/12)
0	1	1	8 bit UART	variable (timer1)
1	0	2	9 bit UART	fixed (xtal/32 or xtal/64)
1	1	3	9 bit UART	variable (timer1)

Mode of operation

- Mode 0 :
  - Serial data enters and exits through RxD
  - TxD outputs the shift clock.
  - 8 bits are transmitted/received(LSB first)
  - The baud rate is fixed a 1/12 the oscillator frequency.

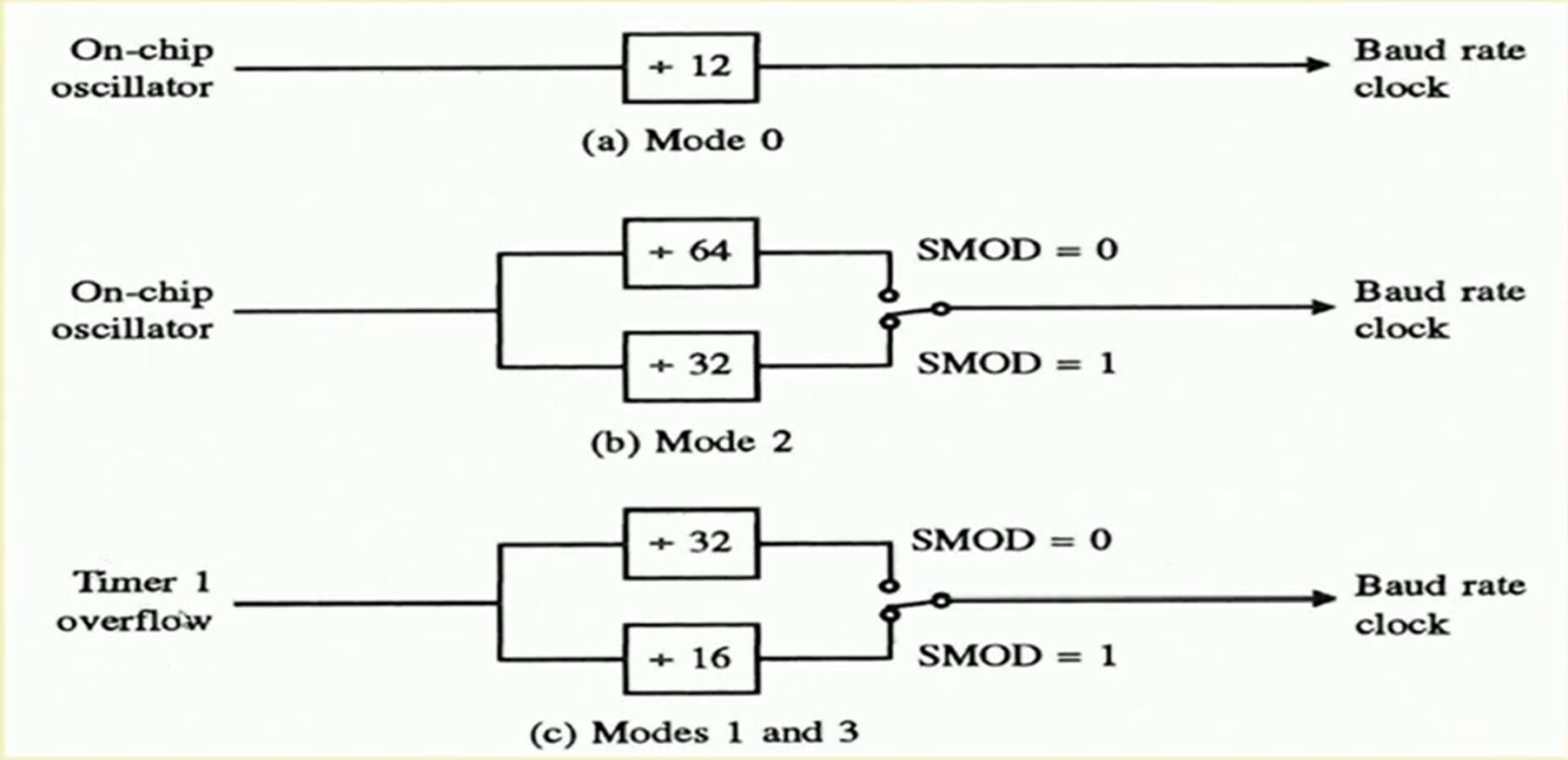
- Application
  - Port expansion



Mode of operation

- Mode 1
  - Ten bits are transmitted (through TxD) or received (through RxD)
  - A start bit (0), 8 data bits (LSB first), and a stop bit (1)
  - On receive, the stop bit goes into RB8 in SCON
  - the baud rate is determined by the Timer 1 overflow rate.
  - Timer1 clock is 1/32 machine cycle (MC=1/12 XTAL)
  - Timer clock can be programmed as 1/16 of machine cycle
  - Transmission is initiated by any instruction that uses SBUF as a destination register.

Mode of operation



Mode of operation

- Mode 2 :
  - Eleven bits are transmitted (through TxD), received (through RxD)
    - A start bit (0)
    - 8 data bits (LSB first)
    - A programmable 9th data bit
    - and a stop bit (1)
  - On transmit, the 9th bit (TB8) can be assigned 0 or 1.
  - On receive, the 9the data bit goes into RB8 in SCON.
  - the 9^{th} can be parity bit
  - The baud rate is programmable to 1/32 or 1/64 the oscillator frequency in Mode 2 by SMOD bit in PCON regist
- Mode 3
  - Same as mode 2
  - But may have a variable baud rate generated from Timer 1.

What is SMOD

- Bit 7 of PCON register
- If SMOD=1 double baud rate
- PCON is not bit addressable
- How to set SMOD

```
Mov a, pcon
Setb acc.7
Mov pcon,a
```

MSB LSB

SMOD	—	—	—	GF1	GF0	PD	IDL

Power control register

MSB LSB

SMOD	—	—	—	GF1	GF0	PD	IDL

BIT	SYMBOL	FUNCTION
PCON.7	SMOD	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
PCON.6	—	Reserved.
PCON.5	—	Reserved.
PCON.4	—	Reserved.
PCON.3	GF1	General-purpose flag bit.
PCON.2	GF0	General-purpose flag bit.
PCON.1	PD	Power-Down bit. Setting this bit activates power-down operation.
PCON.0	IDL	Idle mode bit. Setting this bit activate idle mode operation.

Power control

- A standard for applications where power consumption is critical
- two power reducing modes
  - Idle
  - Power down

Idle mode

- An instruction that sets PCON.0 causes Idle mode
  - Last instruction executed before going into the Idle mode
  - the internal CPU clock is gated off
  - Interrupt, Timer, and Serial Port functions act normally.
  - All of registers , ports and internal RAM maintain their data during Idle
  - ALE and PSEN hold at logic high levels
- Any interrupt
  - will cause PCON.0 to be cleared by HW (terminate Idle mode)
  - then execute ISR
  - with RETI return and execute next instruction after Idle instruction.
- RST signal clears the IDL bit directly

Power-Down Mode

- An instruction that sets PCON.1 causes power dowm mode
- Last instruction executed before going into the power down mode
- the on-chip oscillator is stopped.
- all functions are stopped,the contents of the on-chip RAM and Special Function Registers are maintained.
- The ALE and PSEN output are held low
- The reset that terminates Power Down

Write a program for the 8051 to transfer letter “A” serially at 4800 baud, continuously.

Solution:

```
        MOV  TMOD,#20H  ;timer 1,mode 2(auto reload)
        MOV  TH1,#-6    ;4800 baud rate
        MOV  SCON,#50H  ;8-bit, 1 stop, REN enabled
        SETB TR1        ;start timer 1
AGAIN:  MOV  SBUF,#"A"  ;letter "A" to transfer
HERE:   JNB  TI,HERE    ;wait for the last bit
        CLR  TI         ;clear TI for next char
        SJMP AGAIN      ;keep sending A
```

Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

Solution:

```
        MOV   TMOD,#20H   ;timer 1,mode 2(auto reload)
        MOV   TH1,#-3     ;9600 baud rate
        MOV   SCON,#50H   ;8-bit, 1 stop, REN enabled
        SETB  TR1         ;start timer 1
AGAIN:  MOV   A,#"Y"      ;transfer "Y"
        ACALL TRANS
        MOV   A,#"E"      ;transfer "E"
        ACALL TRANS
        MOV   A,#"S"      ;transfer "S"
        ACALL TRANS
        SJMP  AGAIN       ;keep doing it
;serial data transfer subroutine
TRANS:  MOV   SBUF,A      ;load SBUF
HERE:   JNB   TI,HERE     ;wait for the last bit
        CLR   TI          ;get ready for next byte
         RET
```

Clocks per machine cycle for various 8051 versions

Chip/Maker	Clocks per Machine Cycle
AT89C51 Atmel	12
P89C54X2 Philips	6
DS5000 Dallas Semi	4
DS89C420/30/40/50 Dallas Semi	1

Find the period of the machine cycle (MC) for various versions of 8051, if XTAL=11.0592 MHz.

(a) AT89C51 (b) P89C54X2 (c) DS5000 (d) DS89C4x0

Solution:

(a) 11.0592MHz/12 = 921.6kHz;
MC is 1/921.6kHz = 1.085μs = 1085ns

(b) 11.0592MHz/6 = 1.8432MHz;
MC is 1/1.8432MHz = 0.5425μs = 542ns

(c) 11.0592MHz/4 = 2.7648MHz ;
MC is 1/2.7648MHz = 0.36μs = 360ns

(d) 11.0592MHz/1 = 11.0592MHz;
MC is 1/11.0592MHz = 0.0904μs = 90ns

Instruction	8051	DSC89C4x0
MOV R3,#55	1	2
DEC R3	1	1
DJNZ R2 target	2	4
LJMP	2	3
SJMP	2	3
NOP	1	1
MUL AB	4	9

For an AT8051 and DSC89C4x0 system of 11.0592 MHz, find how long it takes to execute each instruction.

(a) MOV R3,#55 (b) DEC R3 (c) DJNZ R2 target
(d) LJMP (e) SJMP (f) NOP (g) MUL AB

Solution:

	AT8051	*DS89C4x0*
(a)	1×1085ns = 1085ns	2×90ns = 180ns
(b)	1×1085ns = 1085ns	1×90ns = 90ns
(c)	2×1085ns = 2170ns	4×90ns = 360ns
(d)	2×1085ns = 2170ns	3×90ns = 270ns
(e)	2×1085ns = 2170ns	3×90ns = 270ns
(f)	1×1085ns = 1085ns	1×90ns = 90ns
(g)	4×1085ns = 4340ns	9×90ns = 810ns

The following code will continuously send out to port 0 the alternating value 55H and AAH

```
BACK:   MOV     A,#55H
        MOV     P0,A
        ACALL   DELAY
        MOV     A,#0AAH
        MOV     P0,A
        ACALL   DELAY
        SJMP    BACK
```

Port 0 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
        MOV     A,#0FFH         ;A=FF hex
        MOV     P0,A            ;make P0 an i/p port
                                ;by writing it all 1s
BACK:   MOV     A,P0            ;get data from P0
        MOV     P1,A            ;send it to port 1
        SJMP    BACK            ;keep doing it
```

Port 1 is configured first as an input port by writing 1s to it, then data is received from that port and saved in R7 and R5

```
MOV     A,#0FFH     ;A=FF hex
MOV     P1,A        ;make P1 an input port
                    ;by writing it all 1s
MOV     A,P1        ;get data from P1
MOV     R7,A        ;save it to in reg R7
ACALL   DELAY       ;wait
MOV     A,P1        ;another data from P1
MOV     R5,A        ;save it to in reg R5
```

- **Sometimes we need to access only 1 or 2 bits of the port**

```
BACK:   CPL     P1.2            ;complement P1.2
        ACALL   DELAY
        SJMP    BACK

;another variation of the above program
AGAIN:  SETB    P1.2            ;set only P1.2
        ACALL   DELAY
        CLR     P1.2            ;clear only P1.2
        ACALL   DELAY
        SJMP    AGAIN
```

P0	P1	P2	P3	Port Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Write a program to perform the following:
(a) Keep monitoring the P1.2 bit until it becomes high
(b) When P1.2 becomes high, write value 45H to port 0
(c) Send a high-to-low (H-to-L) pulse to P2.3

Solution:

```
        SETB P1.2          ;make P1.2 an input
        MOV  A,#45H        ;A=45H
AGAIN:  JNB  P1.2,AGAIN ; get out when P1.2=1
        MOV  P0,A          ;issue A to P0
        SETB P2.3          ;make P2.3 high
        CLR  P2.3          ;make P2.3 low for H-to-L
```

Example 4-6

A switch is connected to pin P1.7. Write a program to check the status of SW and perform the following:
(a) If SW=0, send letter 'N' to P2
(b) If SW=1, send letter 'Y' to P2
Use the carry flag to check the switch status.

Solution:

```
        SETB P1.7          ;make P1.7 an input
AGAIN:  MOV  C,P1.2        ;read SW status into CF
        JC   OVER          ;jump if SW=1
        MOV  P2,#'N'       ;SW=0, issue 'N' to P2
        SJMP AGAIN         ;keep monitoring
OVER:   MOV  P2,#'Y'       ;SW=1, issue 'Y' to P2
        SJMP AGAIN         ;keep monitoring
```

Example 2-3

Show the status of the CY, AC and P flag after the addition of 9CH and 64H in the following instructions.

```
MOV A, #9CH
ADD A, #64H  ;after the addition A=00H, CY=1
```

Solution:

9C	10011100
+ 64	01100100
100	00000000

CY = 1 since there is a carry beyond the D7 bit

AC = 1 since there is a carry from the D3 to the D4 bi

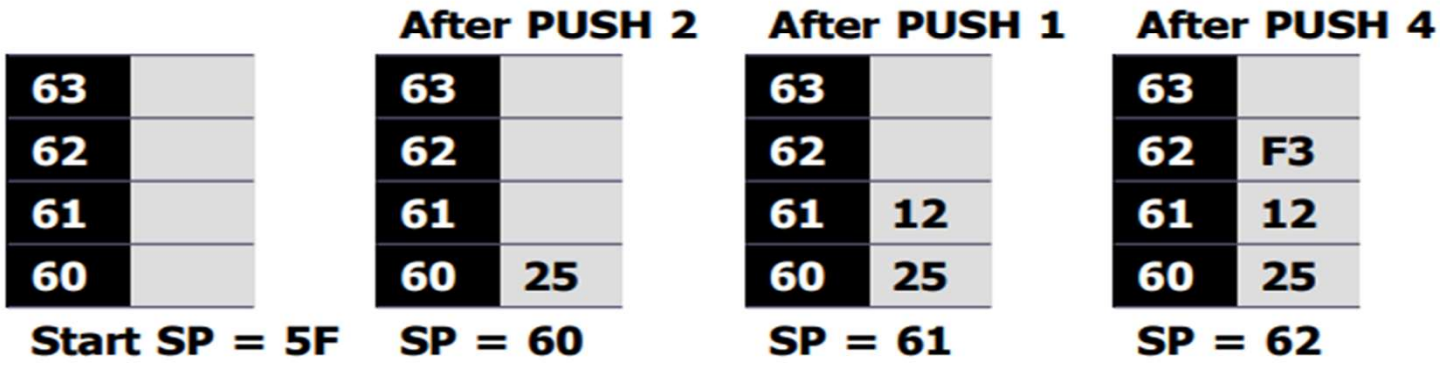
P = 0 since the accumulator has an even number of 1s (it has zero 1s)

Example 2-10

Examining the stack, show the contents of the register and SP after execution of the following instructions. All value are in hex.

```
MOV SP, #5FH     ;make RAM location 60H
                 ;first stack location
MOV R2, #25H
MOV R1, #12H
MOV R4, #0F3H
PUSH  2
PUSH  1
PUSH  4
```

Solution:



Find the sum of the values 79H, F5H, E2H. Put the sum in registers R0 (low byte) and R5 (high byte).

MOV R5,#0

```
        MOV   A,#0       ;A=0
        MOV   R5,A       ;clear R5
        ADD   A,#79H     ;A=0+79H=79H
;       JNC   N_1        ;if CY=0, add next number
;       INC   R5         ;if CY=1, increment R5
N_1:    ADD   A,#0F5H    ;A=79+F5=6E and CY=1
        JNC   N_2        ;jump if CY=0
        INC   R5        ;if CY=1,increment R5 (R5=1)
N_2:    ADD   A,#0E2H    ;A=6E+E2=50 and CY=1
        JNC   OVER       ;jump if CY=0
        INC   R5         ;if CY=1, increment 5
OVER:   MOV   R0,A       ;now R0=50H, and R5=02
```

Example 4-5

A switch is connected to pin P1.7. Write a program to check the status of SW and perform the following:
(a) If SW=0, send letter 'N' to P2
(b) If SW=1, send letter 'Y' to P2

Solution:

```
        SETB P1.7          ;make P1.7 an input
AGAIN:  JB   P1.2,OVER     ;jump if P1.7=1
        MOV  P2,#'N'       ;SW=0, issue 'N' to P2
        SJMP AGAIN         ;keep monitoring
OVER:   MOV  P2,#'Y'       ;SW=1, issue 'Y' to P2
        SJMP AGAIN         ;keep monitoring
```

Example 5-3

Write a program to copy the value 55H into RAM memory locations 40H to 41H using
(a) direct addressing mode, (b) register indirect addressing mode without a loop, and (c) with a loop

Solution:

```
(a)
     MOV A,#55H   ;load A with value 55H
     MOV 40H,A    ;copy A to RAM location 40H
     MOV 41H.A    ;copy A to RAM location 41H
(b)
     MOV A,#55H   ;load A with value 55H
     MOV R0,#40H  ;load the pointer. R0=40H
     MOV @R0,A    ;copy A to RAM R0 points to
     INC R0       ;increment pointer. Now R0=41h
     MOV @R0,A    ;copy A to RAM R0 points to
(c)
        MOV A,#55H     ;A=55H
        MOV R0,#40H    ;load pointer.R0=40H,
        MOV R2,#02     ;load counter, R2=3
AGAIN:  MOV @R0,A      ;copy 55 to RAM R0 points to
        INC R0         ;increment R0 pointer
        DJNZ R2,AGAIN  ;loop until counter = zero
```

Write a program that finds the number of 1s in a given byte.

Solution:

```
        MOV    R1,#0      ;R1 keeps number of 1s
        MOV    R7,#8   ;counter, rotate 8 times
        MOV    A,#97H ;find number of 1s in 97H
AGAIN:  RLC    A          ;rotate it thru CY
        JNC    NEXT       ;check CY
        INC    R1         ;if CY=1, inc count
NEXT:   DJNZ   R7,AGAIN   ;go thru 8 times
```

Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following program

```
        MOV   TMOD,#20H ;T1/8-bit/auto reload
        MOV   TH1,#5    ;TH1 = 5
        SETB  TR1       ;start the timer 1
BACK:   JNB   TF1,BACK  ;till timer rolls over
        CPL   P1.0      ;P1.0 to hi, lo
        CLR   TF1       ;clear Timer 1 flag
        SJMP  BACK      ;mode 2 is auto-reload
```

Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following program

```
        MOV   TMOD,#20H ;T1/8-bit/auto reload
        MOV   TH1,#5    ;TH1 = 5
        SETB  TR1       ;start the timer 1
BACK:   JNB   TF1,BACK  ;till timer rolls over
        CPL   P1.0      ;P1.0 to hi, lo
        CLR   TF1       ;clear Timer 1 flag
        SJMP  BACK      ;mode 2 is auto-reload
```

Solution:

First notice the target address of SJMP. In mode 2 we do not need to reload TH since it is auto-reload. Now (256 - 05) × 1.085 us = 251 × 1.085 us = 272.33 us is the high portion of the pulse. Since it is a 50% duty cycle square wave, the period T is twice that; as a result T = 2 × 272.33 us = 544.67 us and the frequency = 1.83597 kHz