YCCE, Nagpur

MGI

8051 Interrupts & Timers

Dr. P. T. Karule
Professor
Department of Electronics Engineering
Yeshwantrao Chavan College of Engineering, Nagpur (M. S.)
Email: ptkarule@ycce.edu
Website: www.ycce.edu

Interrupts

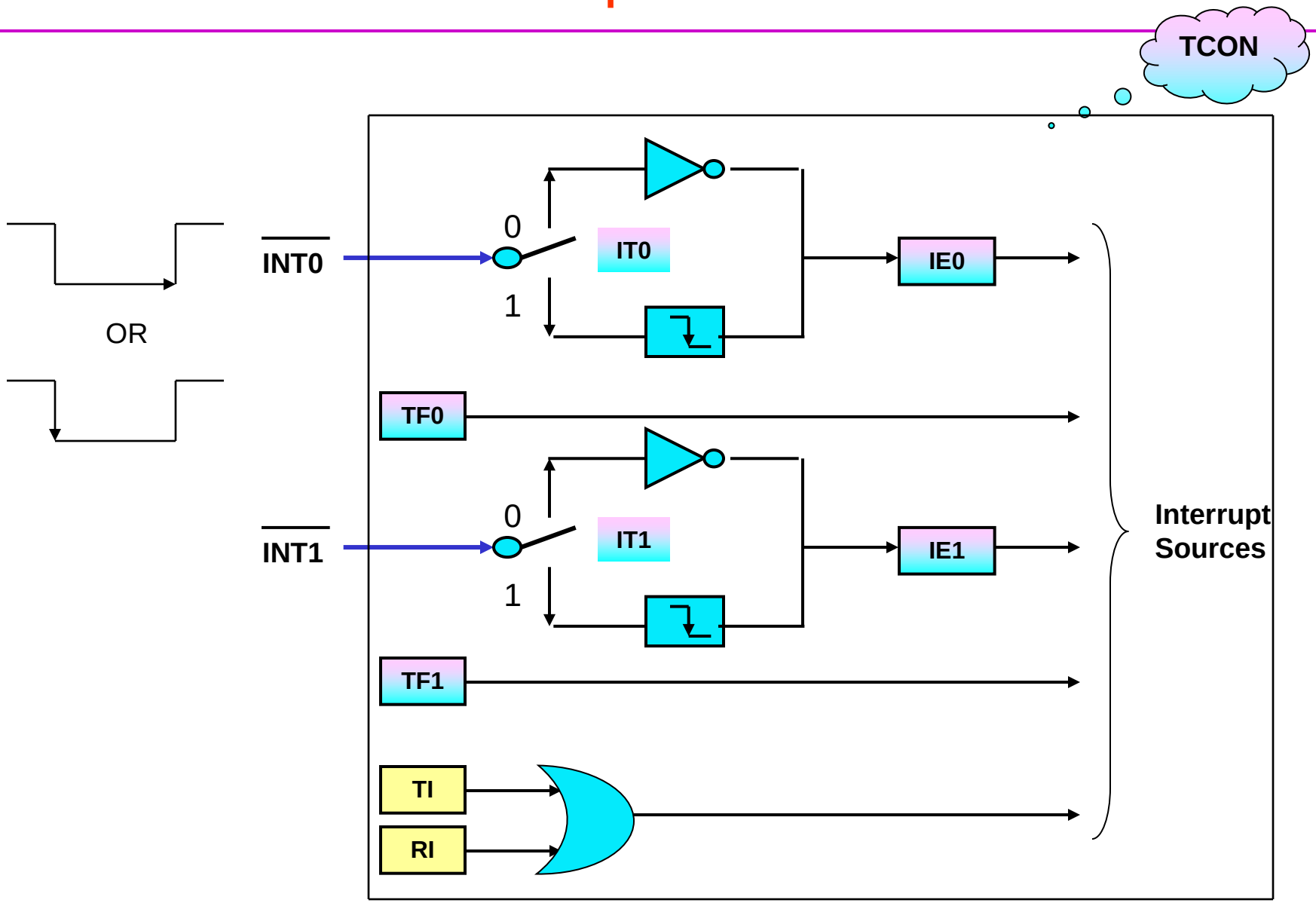
Interrupt sources of 8051

- Total five Interrupt Sources
- Two external interrupt pins (either low level or falling edge triggered)
  - $\overline{\text{INT0}}$
  - $\overline{\text{INT1}}$
- Three internal (Two of timer and one serial)
  - Timer 0 overflow
  - Timer 1 overflow
  - Serial (TI or RI)

Interrupt Vector Table

Interrupt sources	Vector Location Addresses	Priority
$\overline{\text{INT0}}$	0003H	1st
Timer 0 overflow	000BH	2nd
$\overline{\text{INT1}}$	0013H	3rd
Timer 1 overflow	001BH	4th
TI or RI	0023H	5th

Interrupt Circuit



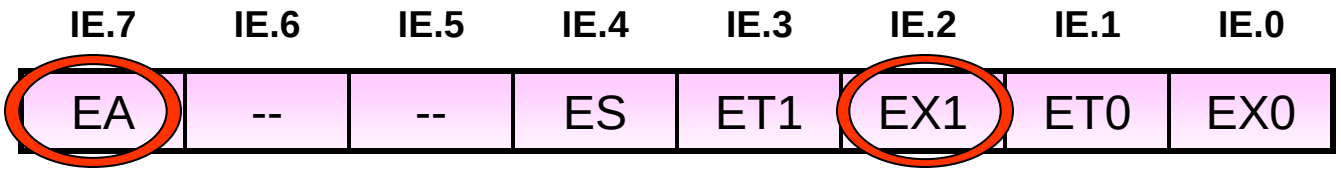
Interrupt Control Registers

SFR	Addr	Bit addr
IE	A8H	A8H to AFH
IP	B8H	B8H to BFH
TCON	88H	88H to 8FH

- Interrupt Enable Register (IE)
  - It is used to enable or disable the individual interrupt.
- Interrupt Priority Register (IP)
  - It is used to set priority of interrupt either high / low
- Timer Control Register (TCON)
  - It is used to set type of trigger for external interrupts.

    (low level or falling edge trigger)
  - It also indicate status of external interrupts & timer overflow

Interrupt Enable Register (IE)

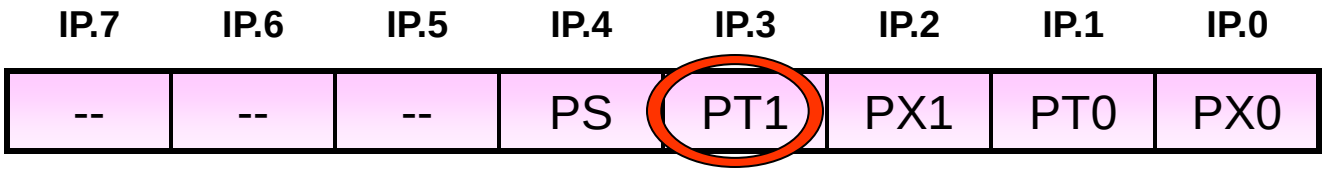


EA (Enable All)	If EA = 0, disable all interrupts. If EA = 1, each interrupt source can be individually enabled or disabled by setting / clearing its control bit.
ES	Enable or disable serial port interrupt. If ES = 0, disable. If ES = 1, enable.
ET1	Enable or disable timer1 overflow interrupt. If ET1 = 0, disable. If ET1 = 1, enable.
EX1	Enable or disable external interrupt1. If EX1 = 0, disable. If EX1 = 1, enable.
ET0	Enable or disable timer0 overflow interrupt. If ET0 = 0, disable. If ET0 = 1, enable.
EX0	Enable or disable external interrupt0. If EX0 = 0, disable. If EX0 = 1, enable.

e.g. Enable external interrupt 1

```
MOV  IE, #10000100B          ; Enable external interrupt 1 (bit EA=1and bit EX1=1)
```

Interrupt Priority Register (IP)



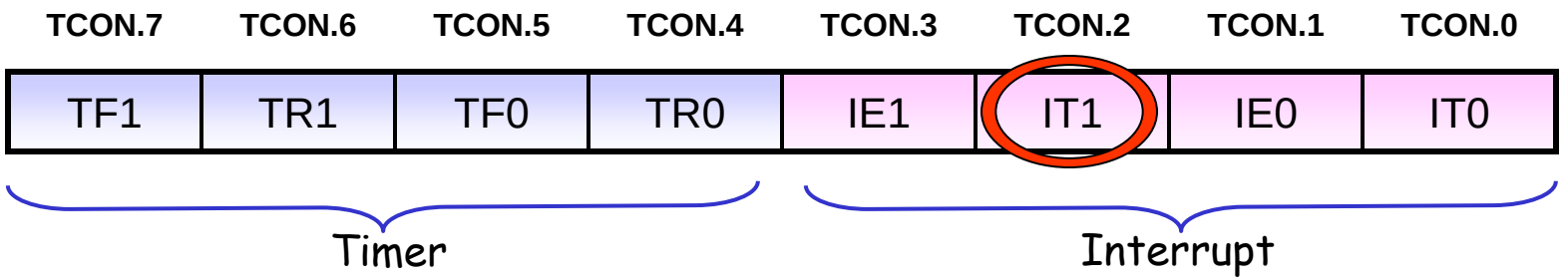
- If bit is low then it corresponds to lower priority
- But if bit is high then it corresponds to higher priority

PS	Priority of serial port interrupt. If PS=0, Serial interrupt is at low priority. If PS=1, Serial interrupt is at high priority.
PT1	Priority of timer1 overflow interrupt.
PX1	Priority of external interrupt 1.
PT0	Priority of timer0 overflow interrupt.
PX0	Priority of external interrupt 0.

e.g. Enable external interrupt 1 & timer 1 overflow interrupt.
Assign high priority to timer1 interrupt.

```
MOV  IE, #10001100B     ; Enable external interrupt 1 (bit EA=1, bit EX1=1 & ET1=1)
SETB  PT1               ; Timer 1 interrupt has high priority
```

TCON



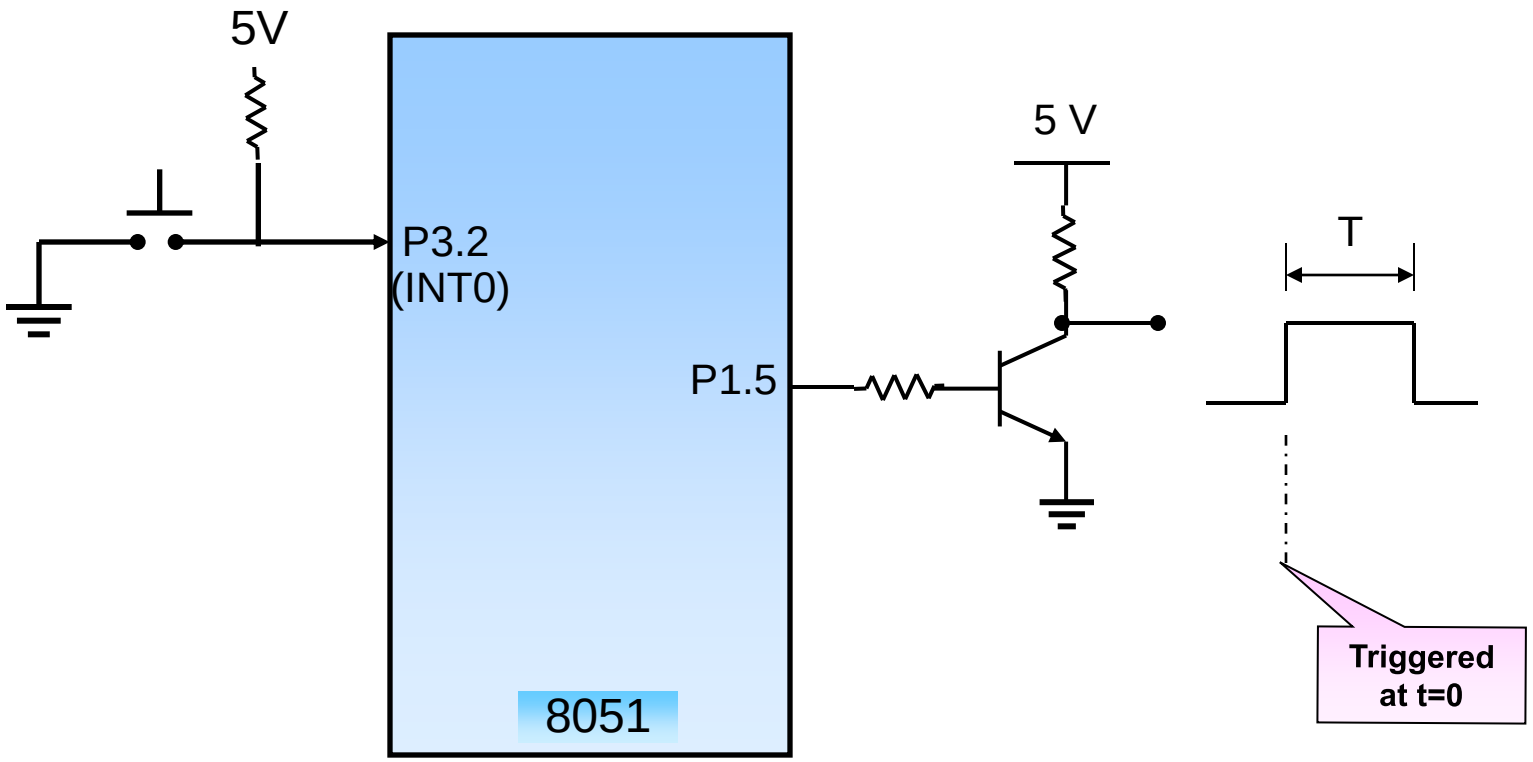
IE1	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

e.g. Write instructions to set interrupt type for external interrupt 1 as falling edge triggered & enable the interrupt

```
SETB  IT1                ; IT1 = 1, falling edge triggered external interrupt 1
MOV  IE, #10000100B      ; Enable external interrupt 1 (bit EA=1and bit EX1=1)
```

Example

- Write program to generate the pulse of width T on P1.5?
  Use external interrupt 0 to apply trigger. i.e. to start the pulse.



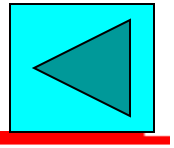
Example (Cont..)

- Program

```
            $MOD51
            ORG 0000H
            SJMP  START
            ORG 0003H
            AJMP INT_EX0
START:    MOV  SP, #40H              ; initialise stack pointer
            CLR  IT0                   ; IT0=0, Low level trigger
            MOV  IE, #10000001B        ; EA =1 & EX0 =1, enable INT0
            CLR  P1.5                  ; P1.5 = 0
      L1:   SJMP L1

; Interrupt Service Routine for INT0

INT_EX0:  SETB  P1.5                   ; P1.5 = 1
            MOV  R0, # COUNT           ; Delay
       L2:  DJNZ  R0, L2
            CLR  P1.5                  ; P1.5 = 0
            RETI                       ; Interrupt return
      END
```



Timer/Counter

Timers & Counters

Features of Timer Register

- 8051 has two timers named as Timer 0 & Timer 1
- Each Timer is a 16 bit register (two 8 bit registers), which can be initialized with any 16 bit no. between 0000H to FFFFH
- Timer register increments after every negative edge of input CLK
- Timer register can only increment, hence it is up counter

THO	TLO

16 bit Timer 0 register

TH1	TL1

16 bit Timer 1 register

Timer Registers	Byte Address
TL0	8AH
TL1	8BH
TH0	8CH
TH1	8DH

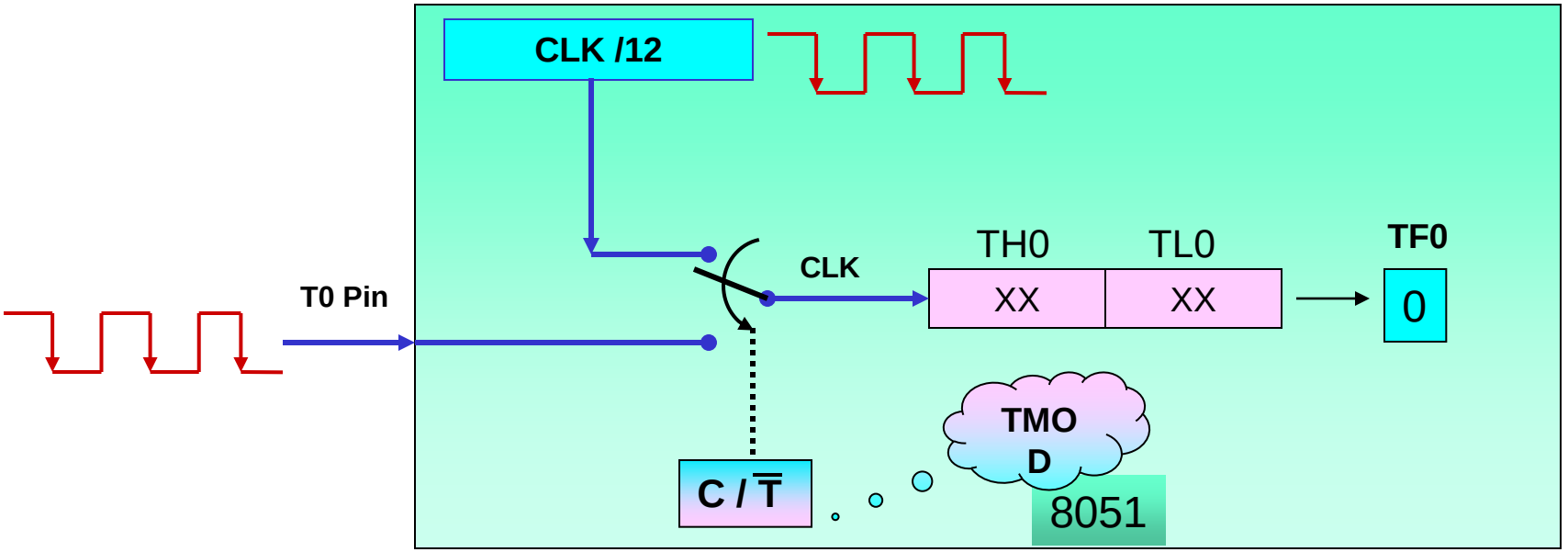
Timer / Counter

- Timer ($C/\overline{T} = 0$)

  Internal clock is used to increment the timer register, timer is incremented after every 12 CLK cycles of internal clock (i.e. One m/c cycle)

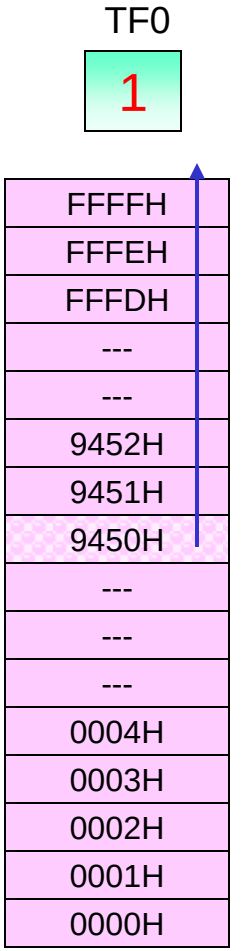
- Counter ($C/\overline{T} = 1$)

  External clock is used to increment the timer register, the register is incremented on a negative edge on input pins T0 / T1.



Timer/Counter operation

- Timer 0 Registers of 8051

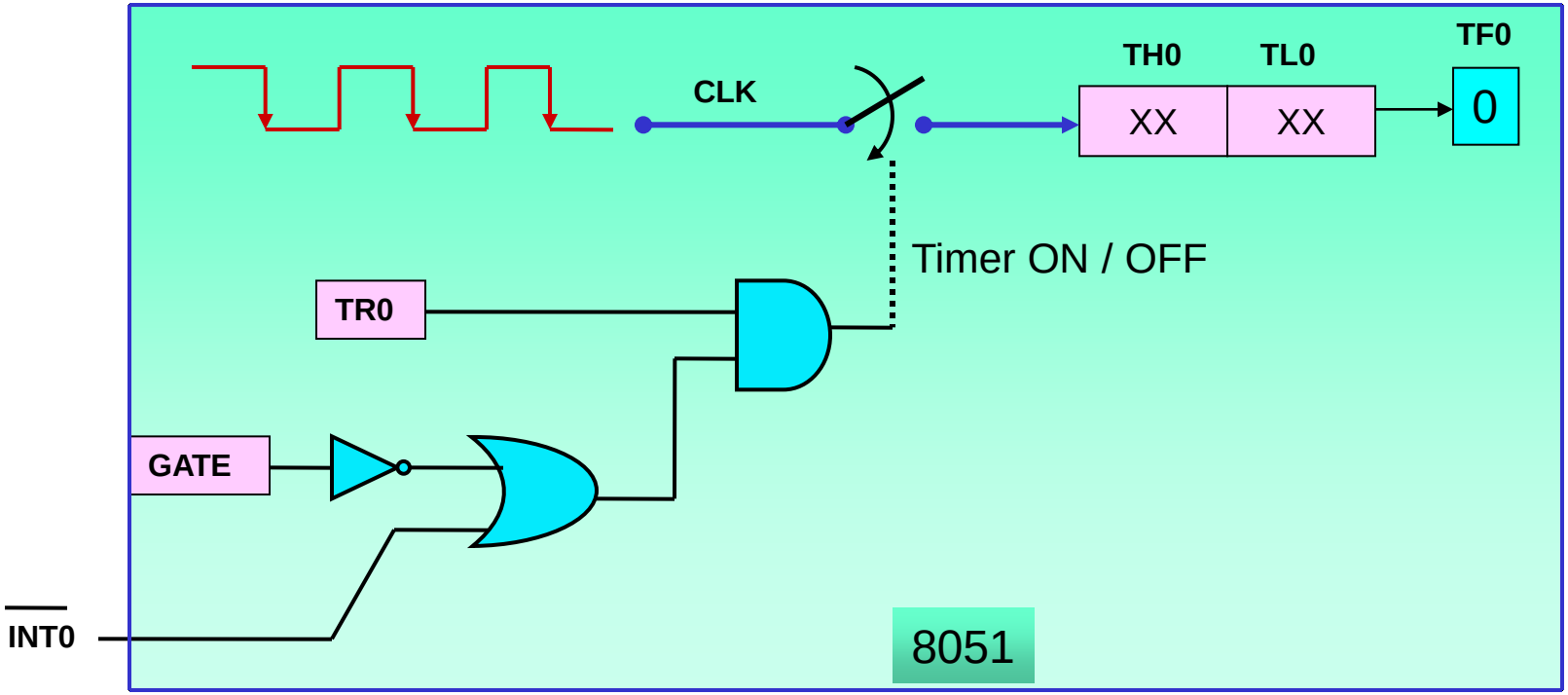


Timer #Value Range

- Initialize timer 0 with N = 9450H

```
MOV  TL0, #50H
MOV  TH0, #94H
```

Timer/Counter control circuit



TR0	GATE	$\overline{\text{INT0}}$	Timer0	
0	X	X	OFF	
1	0	X	ON	S/W Control
1	1	0	OFF	H/W Control
1	1	1	ON	

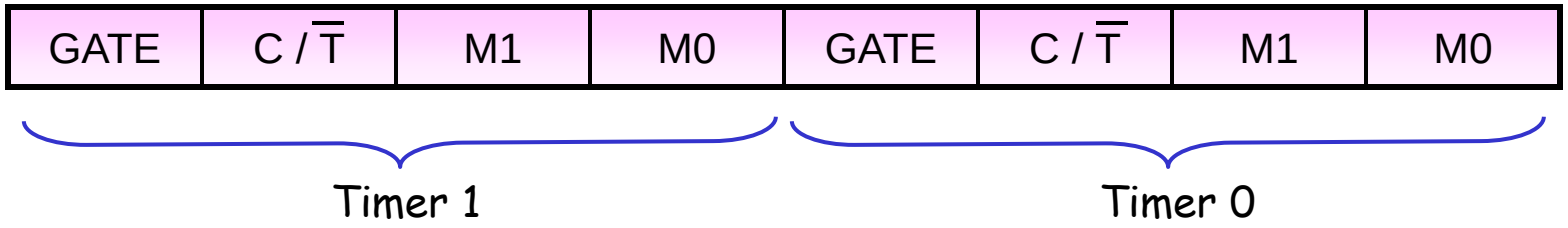
Timer Control Registers

- The operation of timer/counter is determined and controlled by two registers.
  - *TMOD*
  - *TCON*

SFR	Addr	Bit addr
TCON	88H	88H to 8FH
TMOD	89H	--

- TMOD is used to select
  - Timer / Counter (i.e CLK internal / external) (C/T)
  - Mode of operation of timer (M1M0)
    - Mode 0 – 13 bit timer
    - Mode 1 - 16 bit timer
    - Mode 2 - 8 bit timer
    - Mode 3 - Two 8 bit & one 16 bit (3 timers)
  - Timer / Counter control (ON / OFF) (GATE)
    - Hardware control
    - Software control
- TCON is used to select
  - Timer / counter run ON /OFF (TR)
    - Main control for ON / OFF

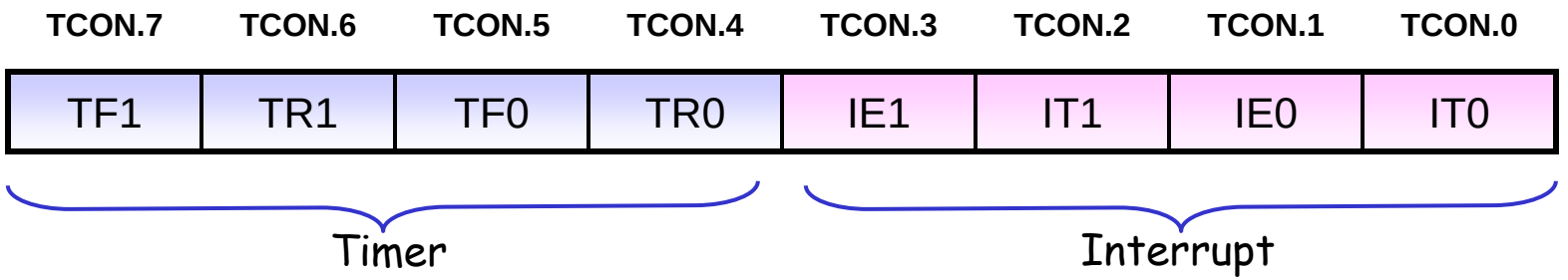
TMOD Register



GATE	If GATE = 0, software controlled timer ON/OFF. If GATE = 1, hardware controlled timer ON/OFF.
C / $\overline{T}$	Timer / Counter selector. If C / T = 0, Timer (internal clock). If C / T = 1, Counter (external clock).

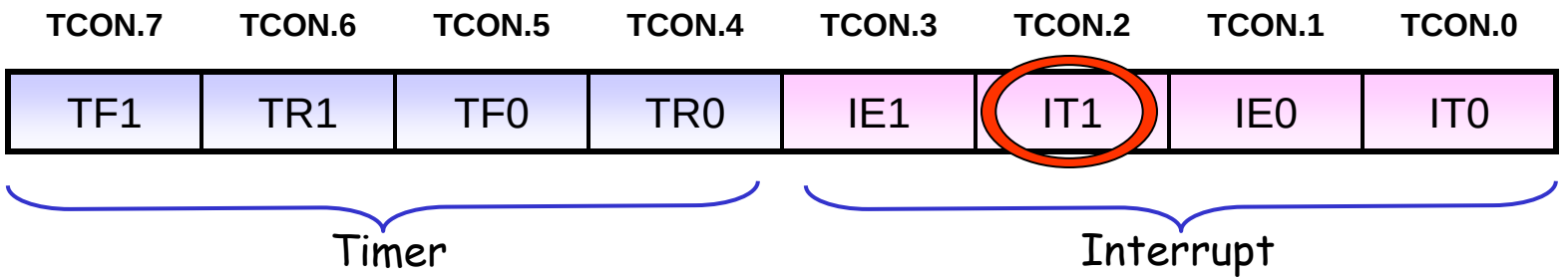
M1	M0	**Operating Mode**
0	0	Mode 0, 13 bit timer/counter
0	1	Mode 1, 16 bit timer/counter
1	0	Mode 2, 8 bit auto-reload timer/counter
1	1	Mode 3, Timer 0 is used as two separate 8 bit timers /counters. TL0 is an 8 bit timer/counter controlled by timer 0 control bits. TH0 is an 8 bit timer/counter controlled by timer 1 control bits.

TCON



TF1	Timer 1 overflow flag. Set by hardware when the Timer1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	Timer 1 run control bit. Set/cleared by software to turn Timer1 ON/OFF.
TF0	Timer 0 overflow flag. Set by hardware when the Timer0 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR0	Timer 0 run control bit. Set/cleared by software to turn Timer0 ON/OFF.

TCON

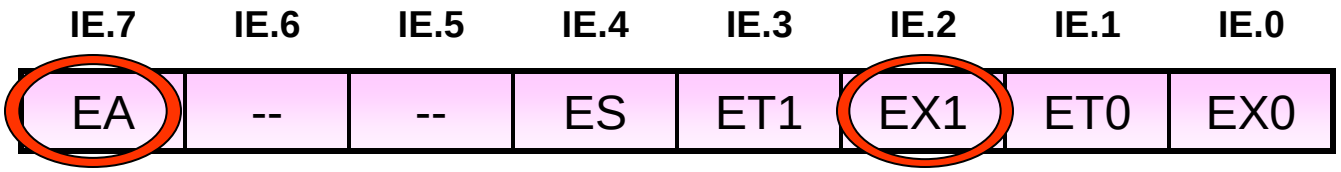


IE1	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

e.g. Write instructions to set interrupt type for external interrupt 1 as falling edge triggered & enable the interrupt

```
SETB  IT1              ; IT1 = 1, falling edge triggered external interrupt 1
MOV  IE, #10000100B    ; Enable external interrupt 1 (bit EA=1and bit EX1=1)
```

Interrupt Enable Register (IE)



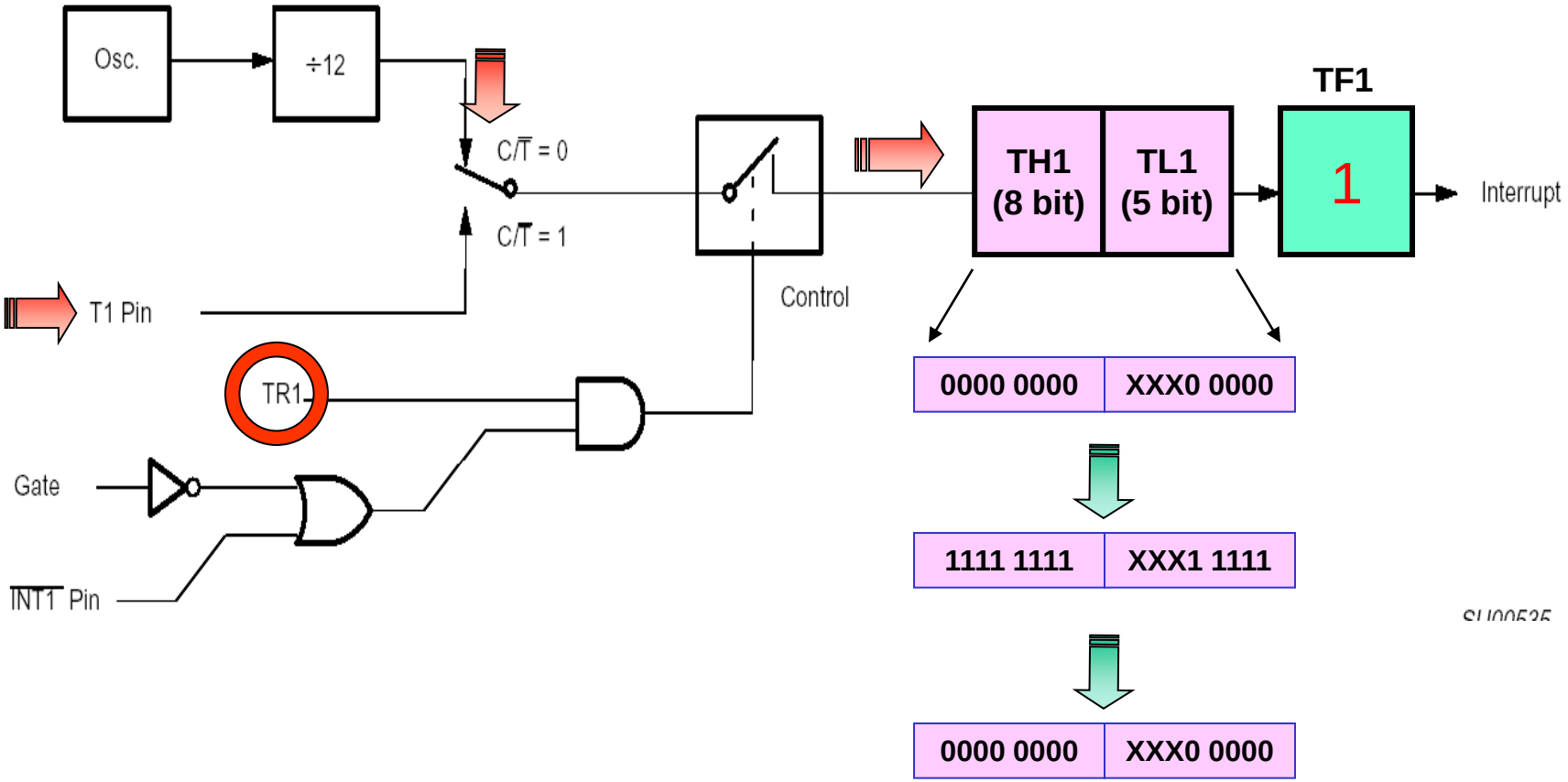
EA (Enable All)	If EA = 0, disable all interrupts. If EA = 1, each interrupt source can be individually enabled or disabled by setting / clearing its control bit.
ES	Enable or disable serial port interrupt. If ES = 0, disable. If ES = 1, enable.
ET1	Enable or disable timer1 overflow interrupt. If ET1 = 0, disable. If ET1 = 1, enable.
EX1	Enable or disable external interrupt1. If EX1 = 0, disable. If EX1 = 1, enable.
ET0	Enable or disable timer0 overflow interrupt. If ET0 = 0, disable. If ET0 = 1, enable.
EX0	Enable or disable external interrupt0. If EX0 = 0, disable. If EX0 = 1, enable.

e.g. Enable external interrupt 1

```
MOV  IE, #10000100B          ; Enable external interrupt 1 (bit EA=1and bit EX1=1)
```

Timer - Mode 0 Operation

- 13 bit timer (0000 to 1FFFH).
- At an overflow of 13 bits, timer overflow flag TF0/TF1 is set.



Programming

- Write program to run Timer 0 as Timer in mode 0 with N = 0000H & Clear bit P1.7 after overflow.

- Program

```
        $MOD 51
        SETB  P1.7              ; Set P1.7
        MOV  TH0, #00H
        MOV  TL0, #00H          ; N = 0000H
        MOV  TMOD,#00H          ; GATE = 0, C/T = 0, M1M0 = 00
        SETB TR0                ; TR0 =1
WAIT:   JNB  TF0, WAIT          ; Wait till TF0 =1 (N = 1FFFH)
        CLR  P1.7               ; Clear P1.7 after overflow
  L1:   SJMP  L1
```

Using interrupt

- Write program to run Timer 0 as Timer in mode 0 with N = 0000H & Clear bit P1.7 after interrupt on overflow.

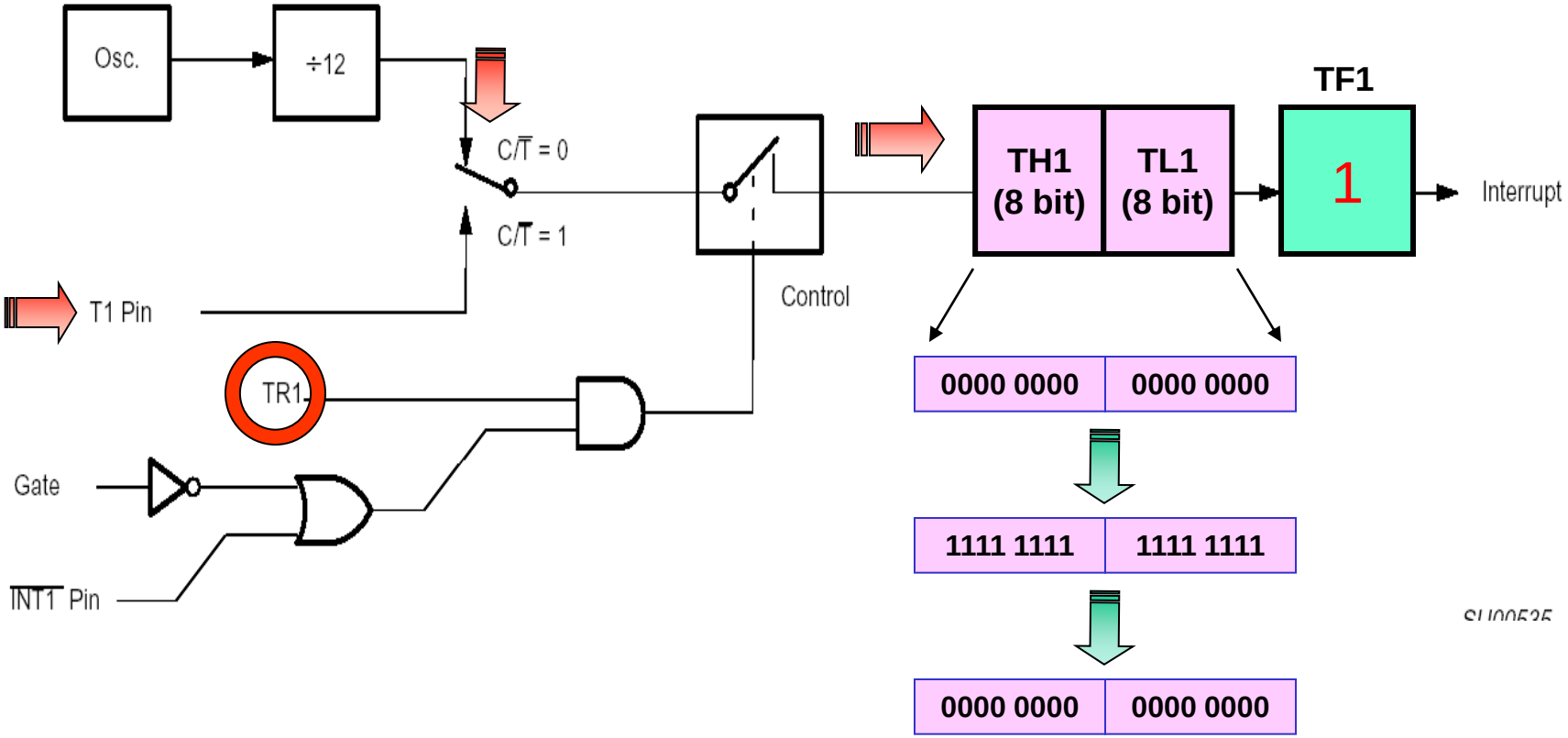
- Main Program

```
                $MOD 51
                ORG  0000H
                SJMP  START

        START:  SETB  P1.7                      ; Set P1.7
                MOV  TH0, #00H
                MOV  TL0, #00H          ; N = 0000H
                MOV  TMOD,#00H          ; GATE = 0, C/T = 0, M1M0 = 00
                SETB TR0                ; TR0 =1, timer0 run.
                MOV  IE, 10000010B      ; Enable timer0 interrupt (EA=1 & ET0=1)
           L1:  SJMP  L1
; Timer 0 overflow interrupt service routine
                ORG 000BH
INT_T0:         CLR  P1.7               ; Clear P1.7
                CLR  TR0                ; Stop timer0
                RETI
```

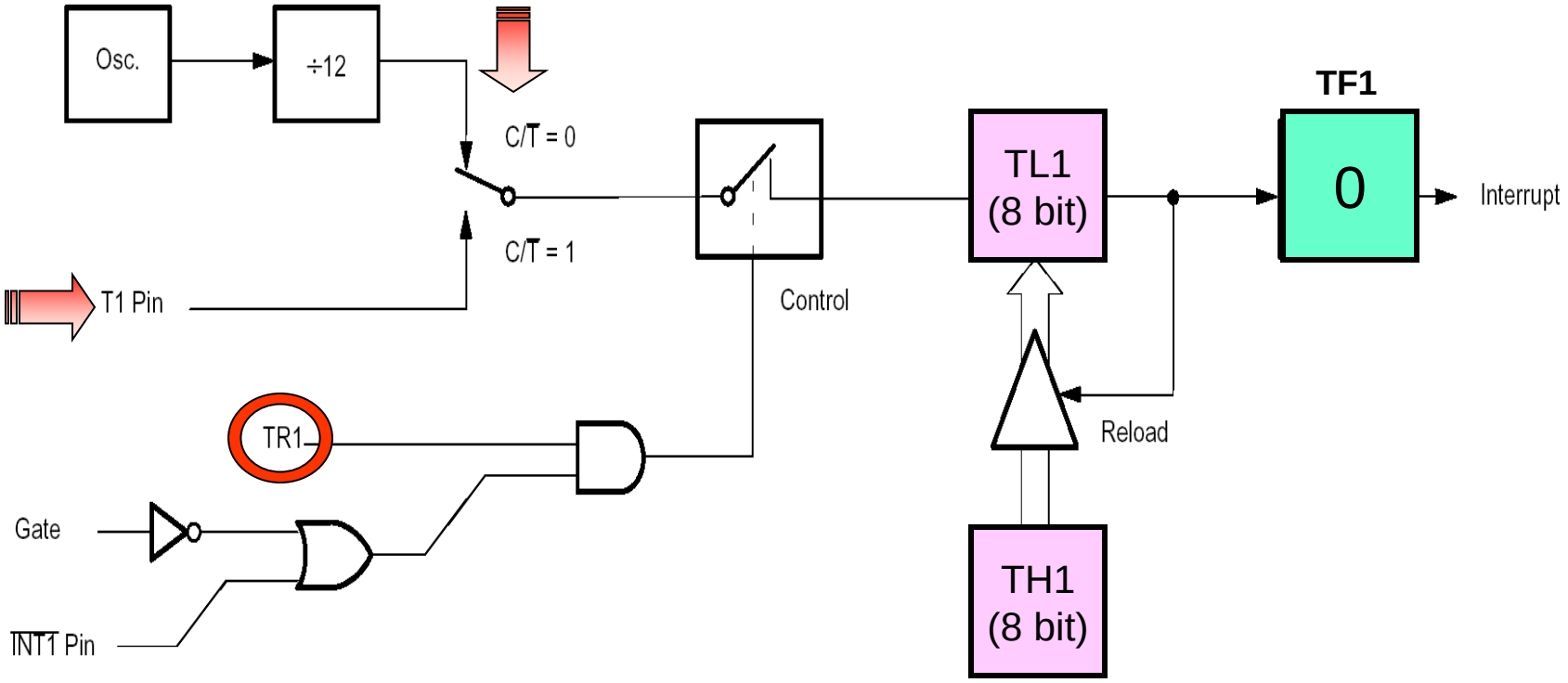
Timer Mode 1

- Timer is used as a 16 bit timer / counter
- After an overflow of 16bits (from 1 to 0), Interrupt flag TFx is set.



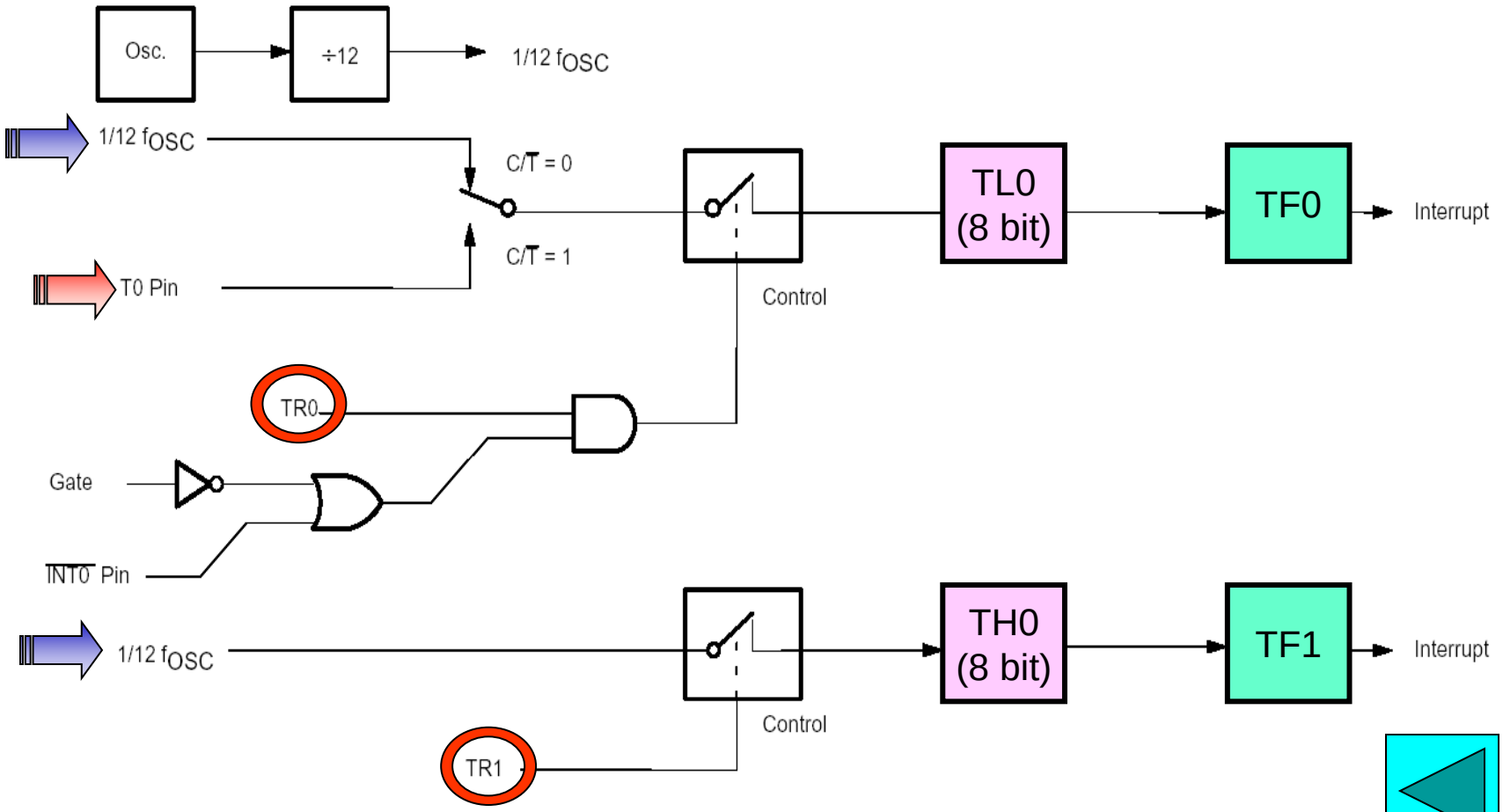
Timer Mode 2 (Reload)

- Timer is used as a 8 bit timer / counter .
- TLx is used as 8 bit counter and THx is used to hold reload value



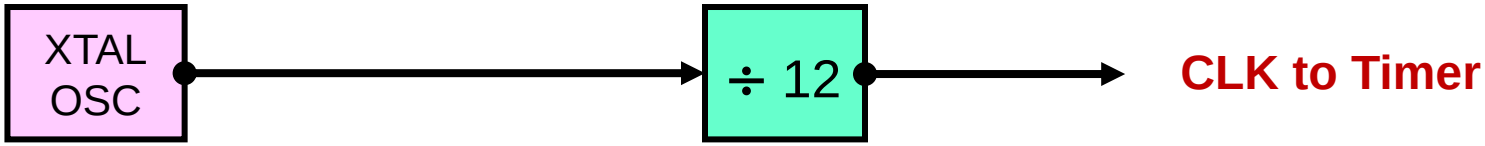
Timer Mode 3

- Timer 0 is split into two 8 bit timers TL0 & TH1
- TL0 is controlled by timer 0 control bits
- TH0 is controlled by timer 1 control bits



CLK input for Timer

- **Find the timer's clock frequency and its period for various 8051-based system, with the crystal frequency 11.0592 MHz when C/T bit of TMOD is 0.**



11.0529 MHz ÷ 12 = 921.6 KHz
T = 1/921.6 KHz = 1.085 μSec

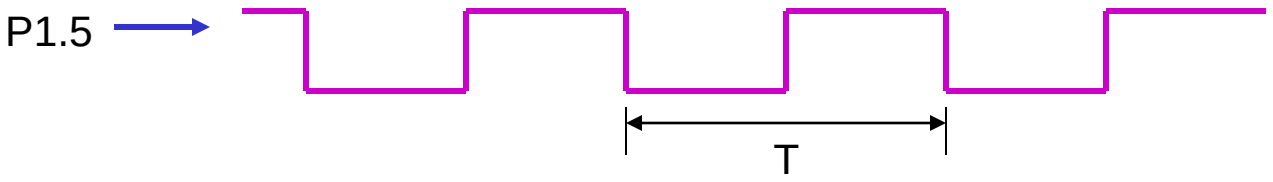
Time Delay

To generate a time delay

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected
2. Load registers TL and TH with initial count value
3. Start the timer
4. Keep monitoring the timer flag (TF) with the JNB TFx,target instruction to see if it is raised
5. Get out of the loop when TF becomes high
6. Stop the timer
7. Clear the TF flag for the next round
8. Go back to Step 2 to load TH and TL again

Ex 1

- Generate a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is to be used to generate the time delay. Analyze the program




```
        MOV TMOD,#01        ;Timer 0, mode 1(16-bit mode)
HERE:   MOV TL0,#0F2H       ;TL0=F2H, the low byte
        MOV TH0,#0FFH       ;TH0=FFH, the high byte
        CPL P1.5            ;toggle P1.5
        ACALL DELAY         ; Delay of T/2
        SJMP HERE
```

Ex 1…

In the above program the steps.

- TMOD is loaded.
- FFF2H is loaded into TH0-TL0.
- P1.5 is toggled for the high and low portions of the pulse.

```
DELAY: SETB TR0          ;start the timer 0
AGAIN: JNB TF0,AGAIN     ;monitor timer flag 0 until it rolls over
       CLR TR0           ;stop timer 0
       CLR TF0           ;clear timer 0 flag
       RET
```

Find Delay Time = T/2 ?

FFF2	FFF3	FFF4	FFF5
TF=0	TF=0	TF=0	TF=0

Find Delay Time = 14 * 1 usec

FFFF	0000
TF=0	TF=1

Ex 2

- Write assembly language program to generate delay of 25 msec using Timer 0 in mode 1.

N = 65535 – 25000 = 40535 = 9657H

```
        MOV TMOD,#01        ;Timer 0, mode 1(16-bit mode)
        MOV TL0,#57H        ;TL0=57H, the low byte
        MOV TH0,#96H        ;TH0=96H, the high byte
        SETB TR0            ;start the timer 0
AGAIN:  JNB TF0,AGAIN       ;monitor timer flag 0 until it rolls over
        CLR TR0             ;stop timer 0
        CLR TF0             ;clear timer 0 flag
L1:     SJMP L1 / RET
```

Thank You!!