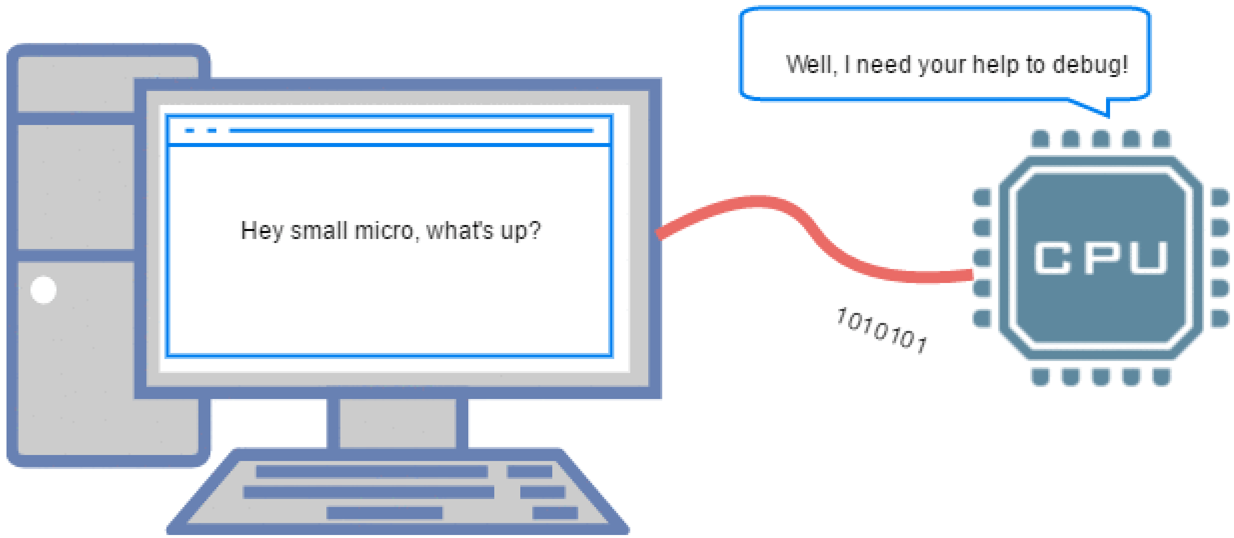
Microprocessors & Microcontrollers

Timers in 8051



Dr. Ujjaval Patel

8051 TIMERS

Timer Registers

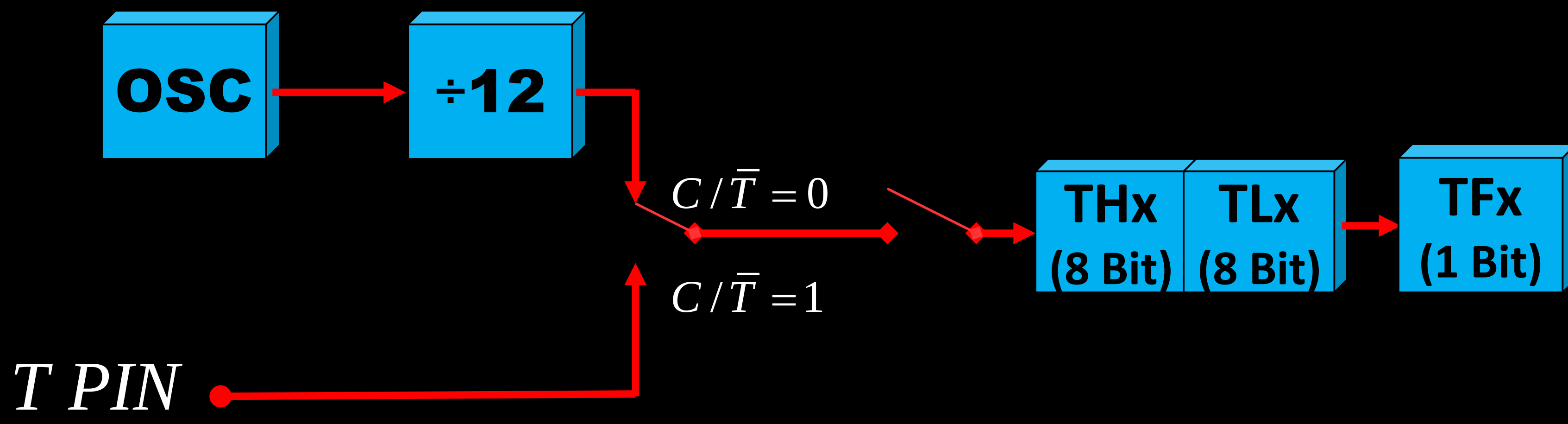
TH0								TL0							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

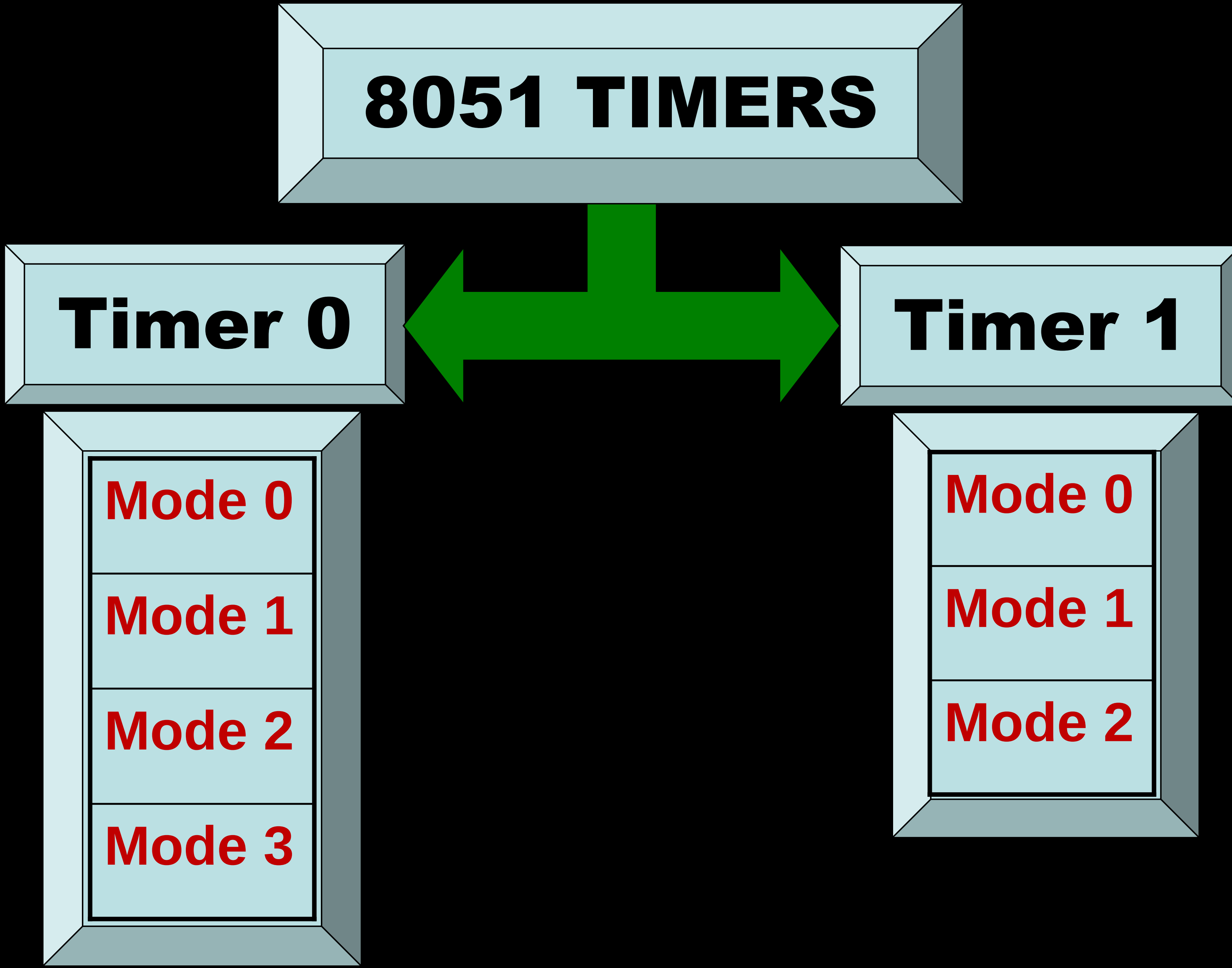
Timer 0

TH1								TL1							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Timer 1

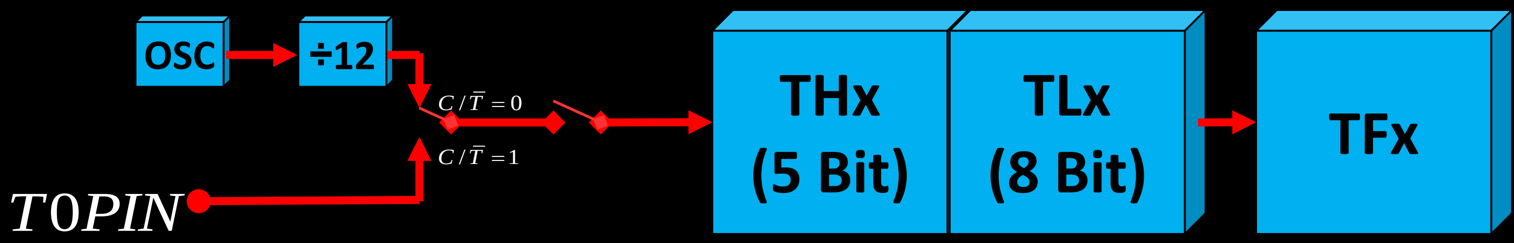
8051 Timer/Counter




8051 TIMERS
Timer 0
Timer 1
Mode 0
Mode 1
Mode 2
Mode 3
Mode 0
Mode 1
Mode 2

TIMER x – Mode 0

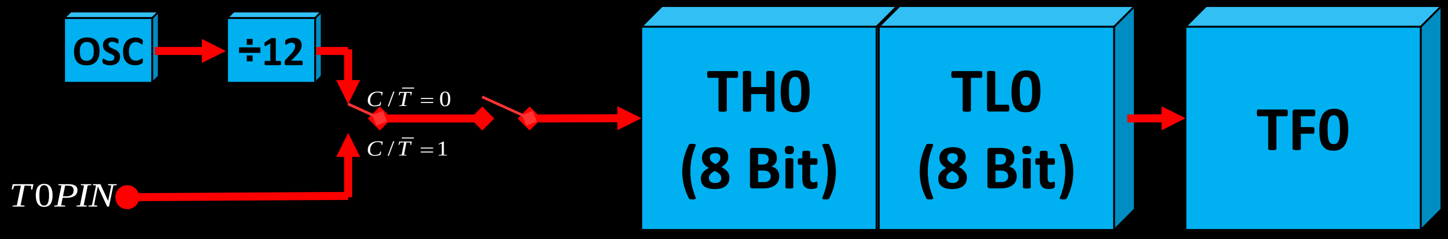
13 Bit Timer / Counter



Maximum Count = 1FFFh (0001111111111111)

TIMER 0 – Mode 1

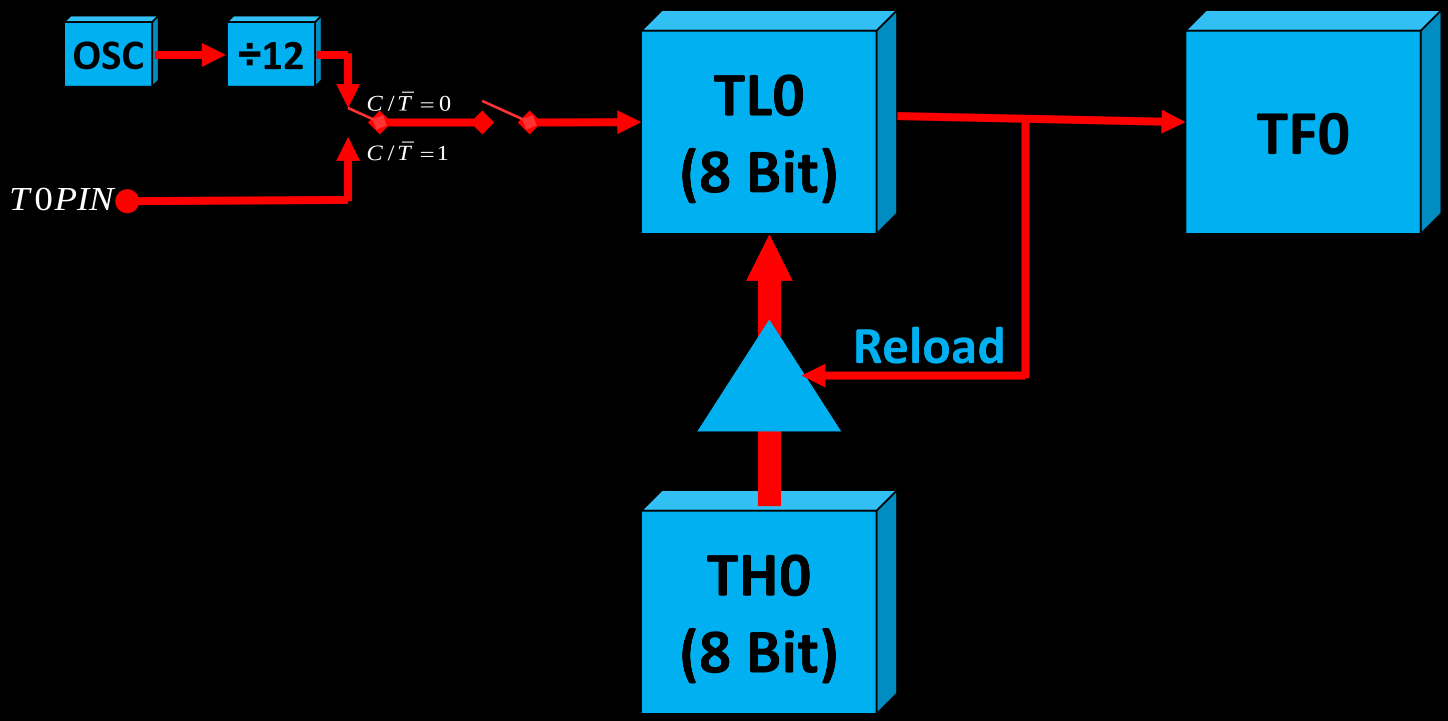
16 Bit Timer / Counter



Maximum Count = FFFFh (1111111111111111)

TIMER 0 – Mode 2

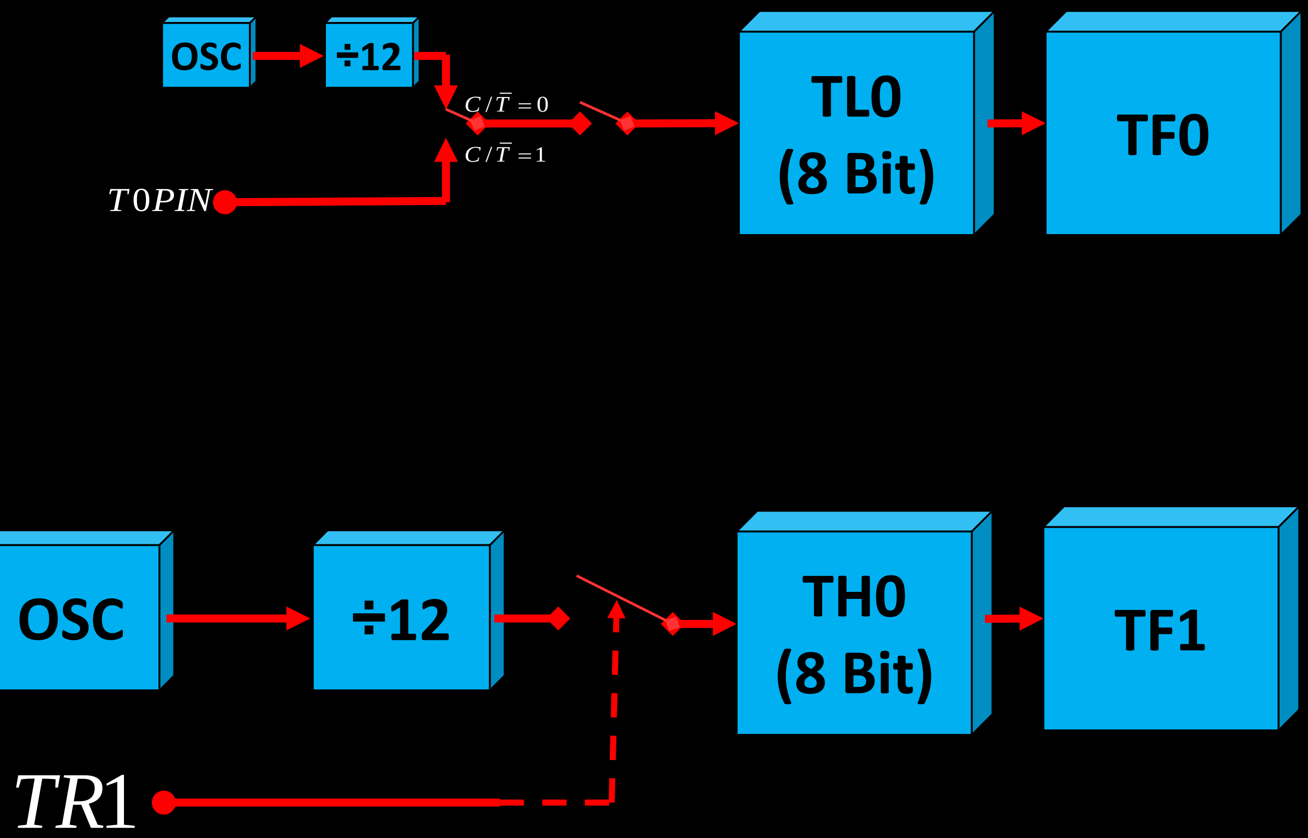
8 Bit Timer / Counter with AUTORELOAD



Maximum Count = FFh (11111111)

TIMER 0 – Mode 3

Two - 8 Bit Timer / Counter



TMOD Register

MSB							LSB
GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

89H

-------------- TIMER 1 -------------- -------------- TIMER 0 --------------

The TMOD byte is not bit addressable.

M1	M0	Operation
0	0	8048 8-bit timer TLx serves as 5-bit prescaler
0	1	16-bit timer/counter. THx and TLx are cascaded. No prescaler
1	0	8-bit autoreload timer/counter. THx contents loaded into TLx when it overflows
1	1	TL0 is 8-bit counter controlled by timer 0 control bits. TH0 is 8-bit timer controlled by timer 1 control bits
1	1	Timer 1 off

TMOD Register

MSB							LSB
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

89H

-------------- TIMER 1 -------------- -------------- TIMER 0 --------------

GATE:

When set, timer/counter x is enabled, **if** INTx pin is high and TRx is set.
When cleared, timer/counter x is enabled, **if** TRx bit set.

C/T*:

When set, counter operation (input from Tx input pin).
When cleared, timer operation (input from internal clock).

TMOD Register

MSB							LSB
GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

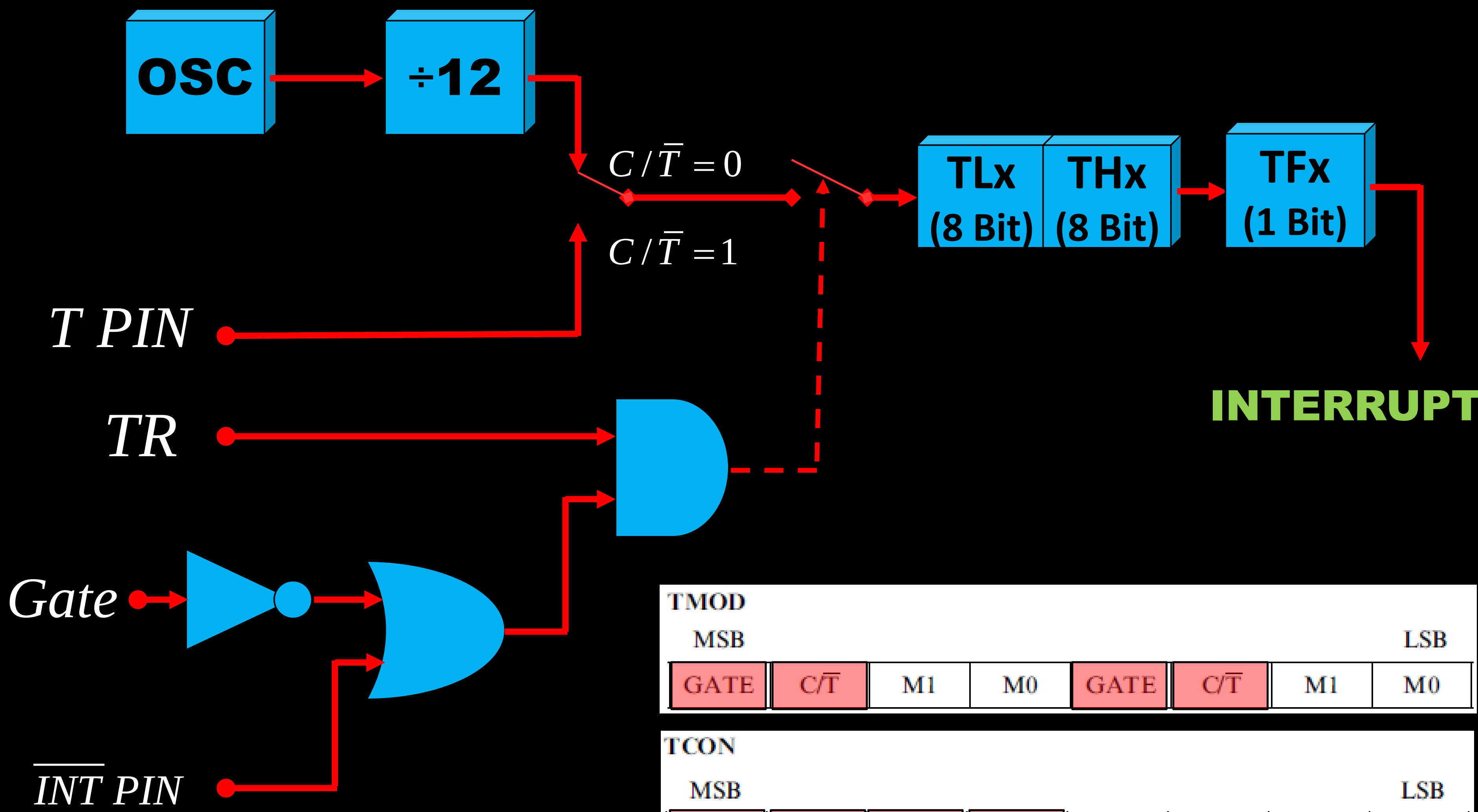
89H

-------------- TIMER 1 -------------- -------------- TIMER 0 ---------------

TCON Register

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

8051 Timer/Counter



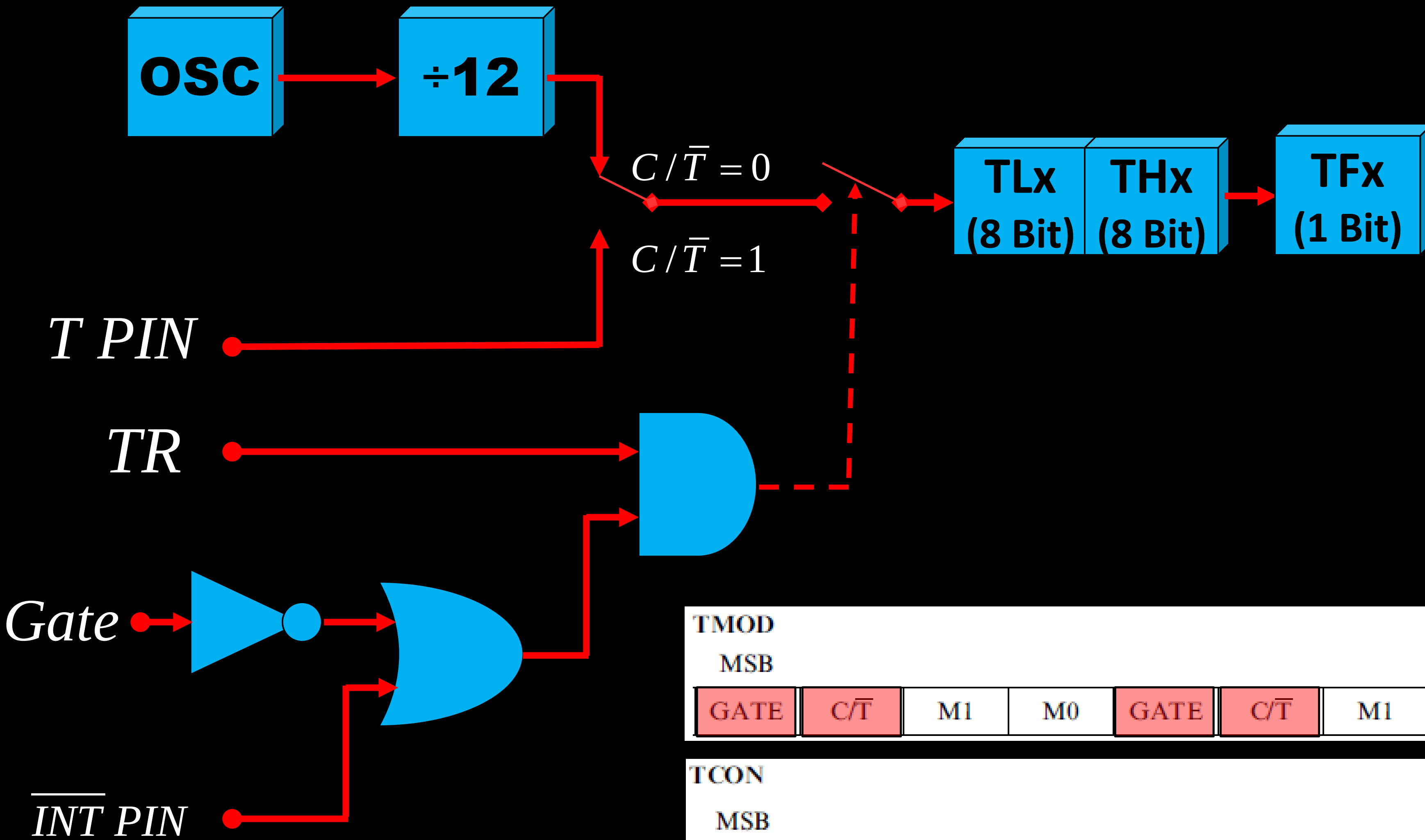
TMOD

MSB							LSB
GATE	C/T	M1	M0	GATE	C/T	M1	M0

TCON

MSB							LSB
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

8051 Timer/Counter



TMOD

MSB — LSB

GATE	C/T̄	M1	M0	GATE	C/T̄	M1	M0

TCON

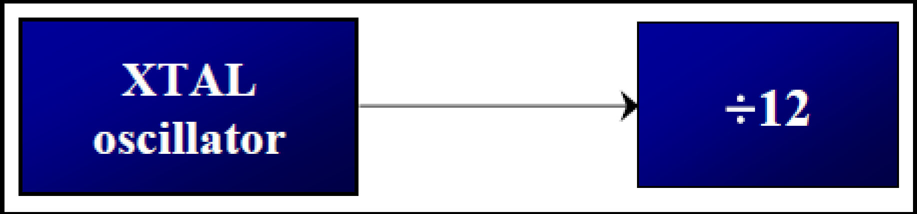
MSB — LSB

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Programming Timers

- 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.

- **Solution:**



$1/12 \times 11.0529$ MHz = 921.6 MHz;

T = 1/921.6 kHz = 1.085 us

How to use Timers ?

1. Configure timer in TMOD Register

MSB LSB

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

89H

-------------- TIMER 1 -------------- -------------- TIMER 0 --------------

Configure Timers

- **Example:** Indicate which mode and which timer are selected for each of the following.

  (a) MOV TMOD, #01H

  (b) MOV TMOD, #20H

  (c) MOV TMOD, #12H

MSB — LSB

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
TIMER 1				TIMER 0			

89H

Steps to use timer in any mode:

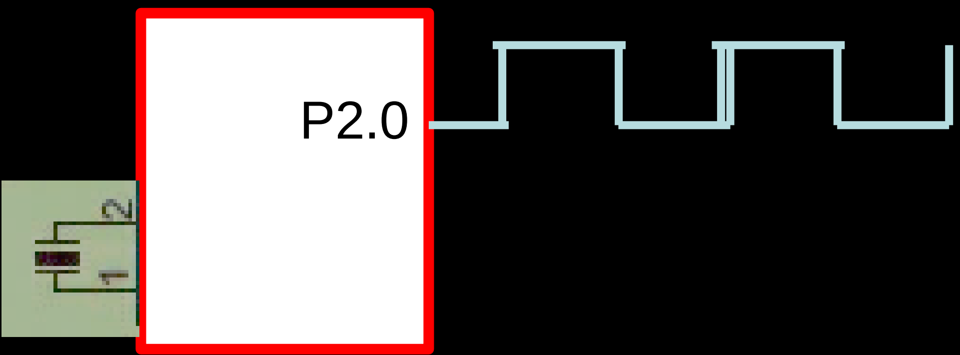
1. Configure timer in TMOD Register
2. Load the timers:
   - Count = Tr / Tm
   - Value to be loaded in the timer = n – Count

   Where n = 256 for Mode 2 &

   n = 65536 FOR Mode 1
3. Start timer by making TRx = 1
4. Monitor TFx bit, when TFx = 1, required time delay is over.
5. Stop timer

Programming Timers

- Write a program to generate a square wave with a frequency of 10 KHz on P2.0.



1) Fr = 10 KHz
   - T = 1/f = 0.1 ms
   - Ton = Toff = Tr = 0.05ms=50 µS
2) Fcry = 11.0592 MHz
   - Fm = Fcry / 12 = 921.6 kHz
   - Tm = 1/Fm= 1.085 µS
3) Count = Tr /Tm = 46 < 256
4) Tx = n – Count
   - Tx = 256 – 46 = 210 = D2h

How to use Timers ?

1. Configure timer in TMOD Register

2. Load the timers:

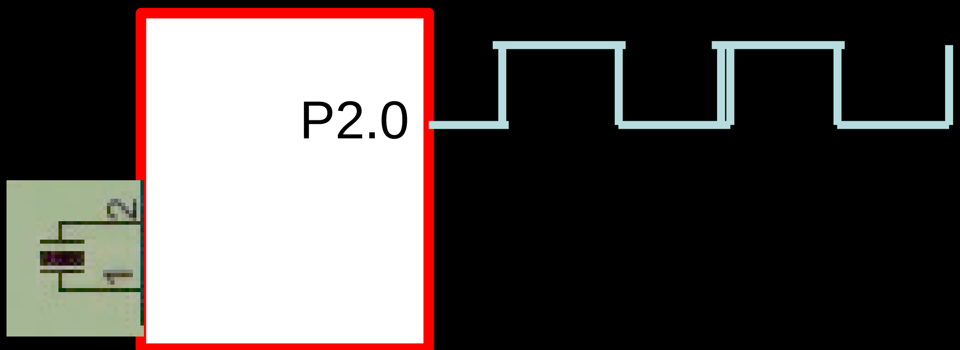
3. Start timer by making TRx = 1
4. Monitor TFx bit, when TFx = 1, required time delay is over.

5. Stop timer

Program…??

Programming Timers

- Write a program to generate a square wave with a frequency of 50 Hz on P2.0.



1) Fr = 50 Hz
   - T = 1/f = 0.02 s = 20 ms
   - Ton = Toff = Tr = 10 ms
2) Fcry = 11.0592 MHz
   - Fm = Fcry / 12 = 921.6 kHz
   - Tm = 1/Fm= 1.085 µS
3) Count = Tr /Tm = 9216 > 256
4) Tx = n – Count
   - Tx =65536–9216 = 56320
   - Tx = | DC | 00 |

THx TLx

How to use Timers ?

1. Configure timer in TMOD Register

2. Load the timers:

3. Start timer by making TRx = 1
4. Monitor TFx bit, when TFx = 1, required time delay is over.

5. Stop timer

Program…??

Counter Operation

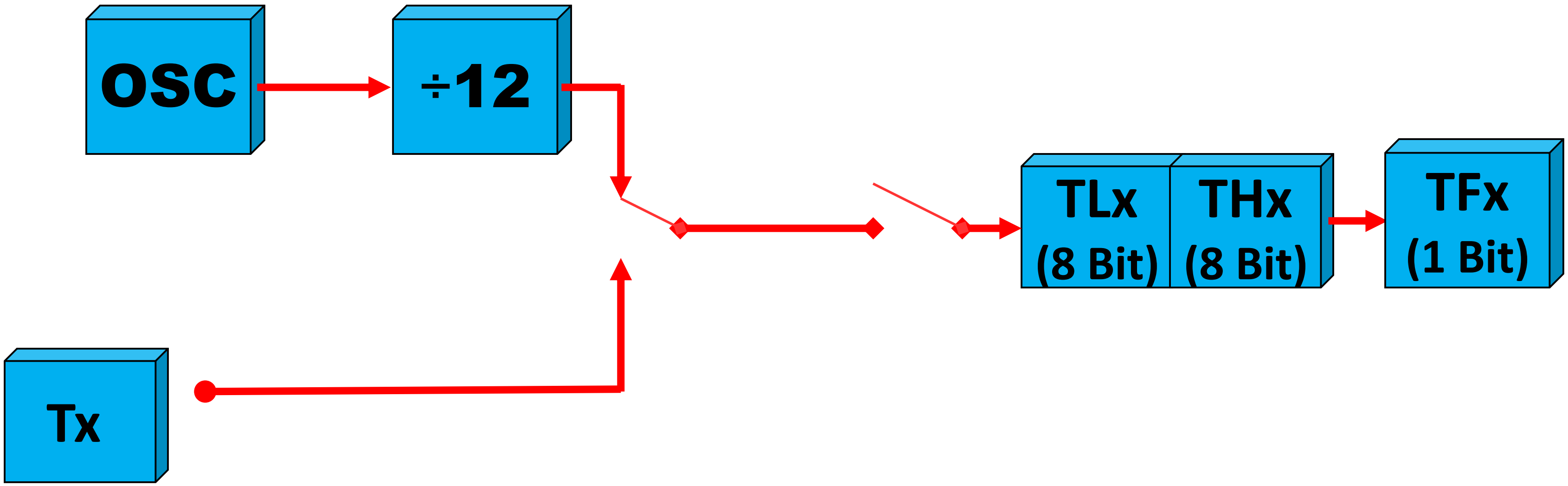
- These timers can also be used as counters counting events happening outside the 8051.

- As far as the use of a timer/counter as an event counter is concerned ,everything that we have talked about in the last section also applies to programming it as a counter ,except the source of the frequency.

- When used as a timer ,the 8051's crystal is used as the source of the frequency.

- However ,when used as a counter ,it is a pulse outside of the 8051 that increments the TH,TL registers.

Counter operation



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			

How to use Timers as counter

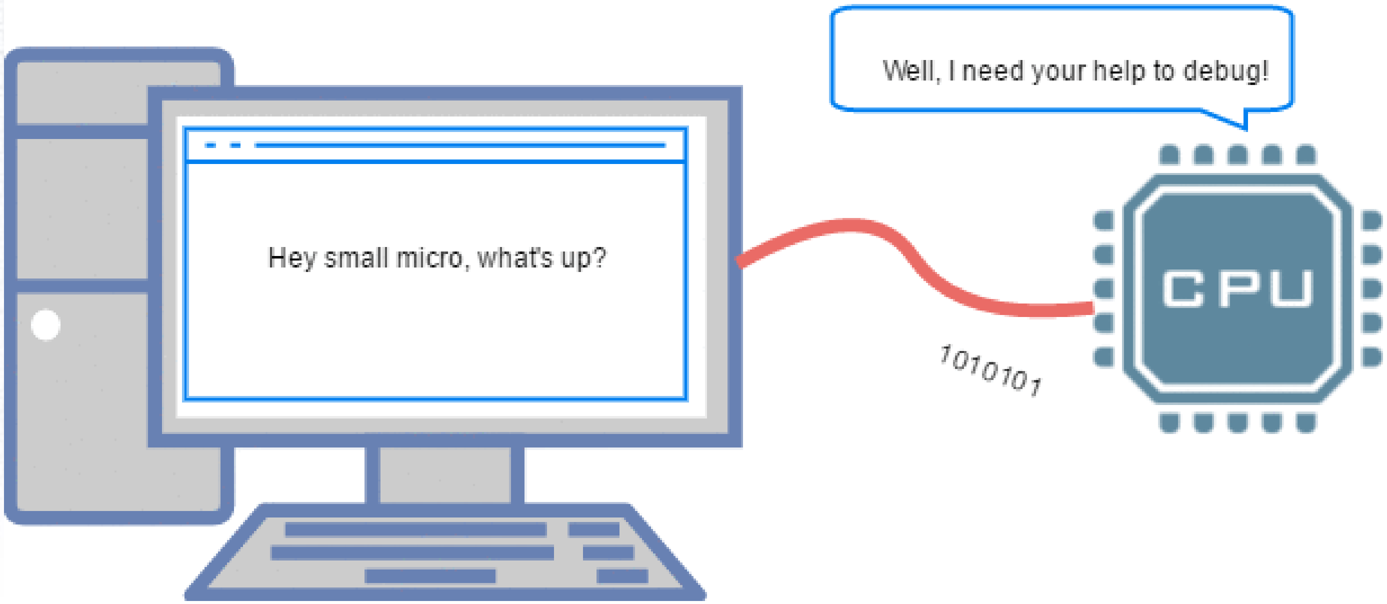
1. Configure timer in TMOD Register: C/T = 1

2. Load the timers:

3. Start timer by making TRx = 1

4. **Read the count value** from timer register and send it for further processing.

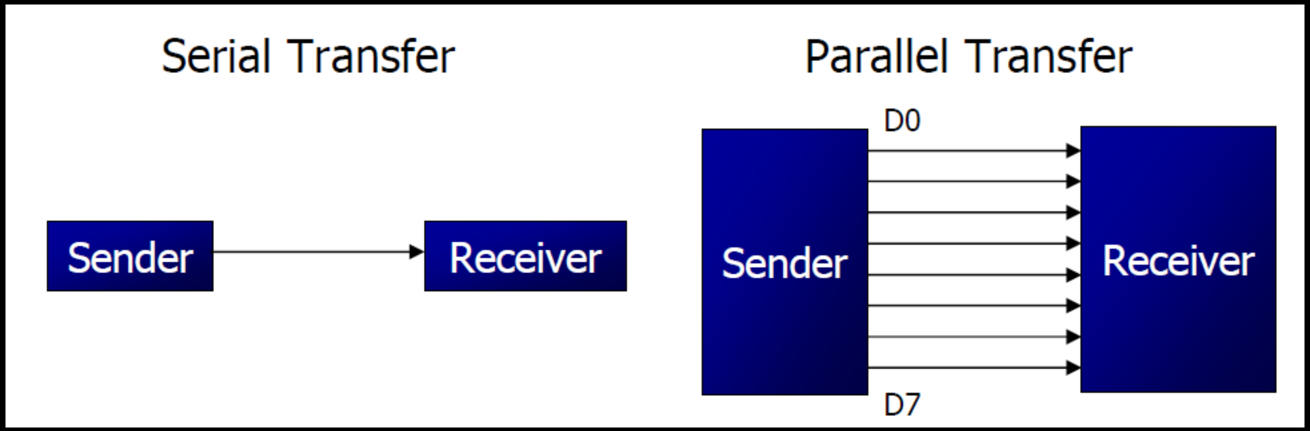
Serial Communication in 8051



Dr. Ujjaval Patel

Basics of Serial Communication

- Computers transfer data in **two** ways:
  - **Parallel:** Often 8 or more lines (wire conductors) are used to transfer data to a device that is only a few feet away.
  - **Serial:** To transfer to a device located many meters away, the serial method is used. The data is sent one bit at a time.

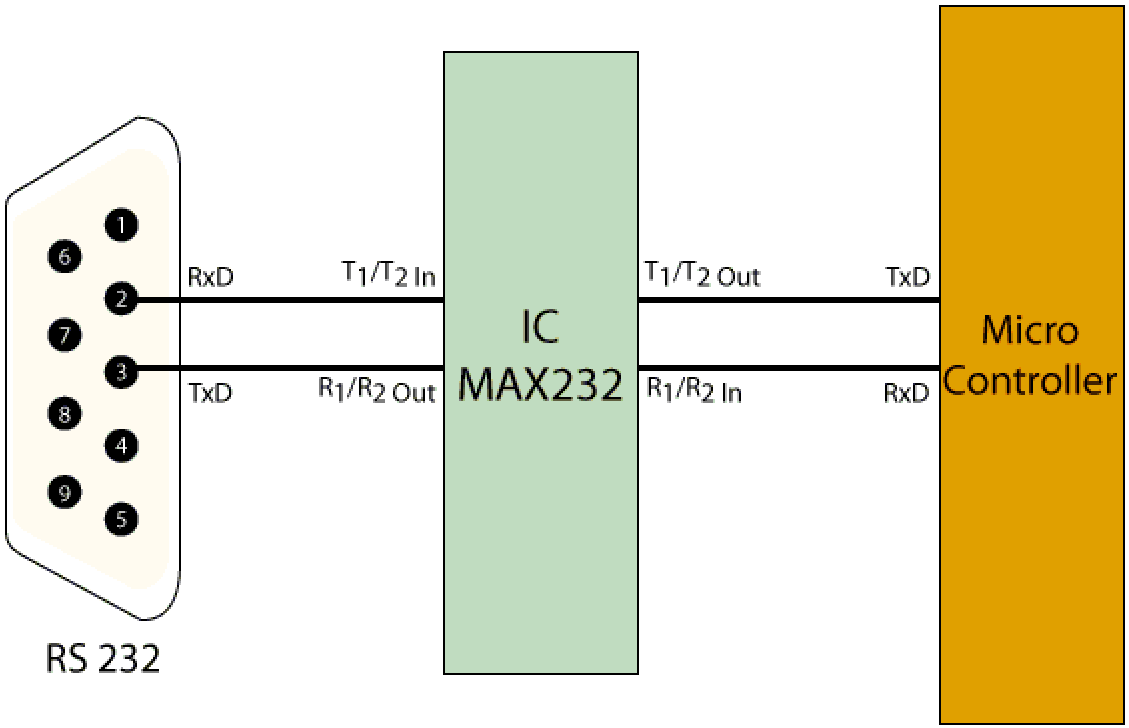


Basics of Serial Communication

- Serial data communication uses **two** methods
  - **Synchronous** method transfers a block of data at a time
  - **Asynchronous** method transfers a single byte at a time

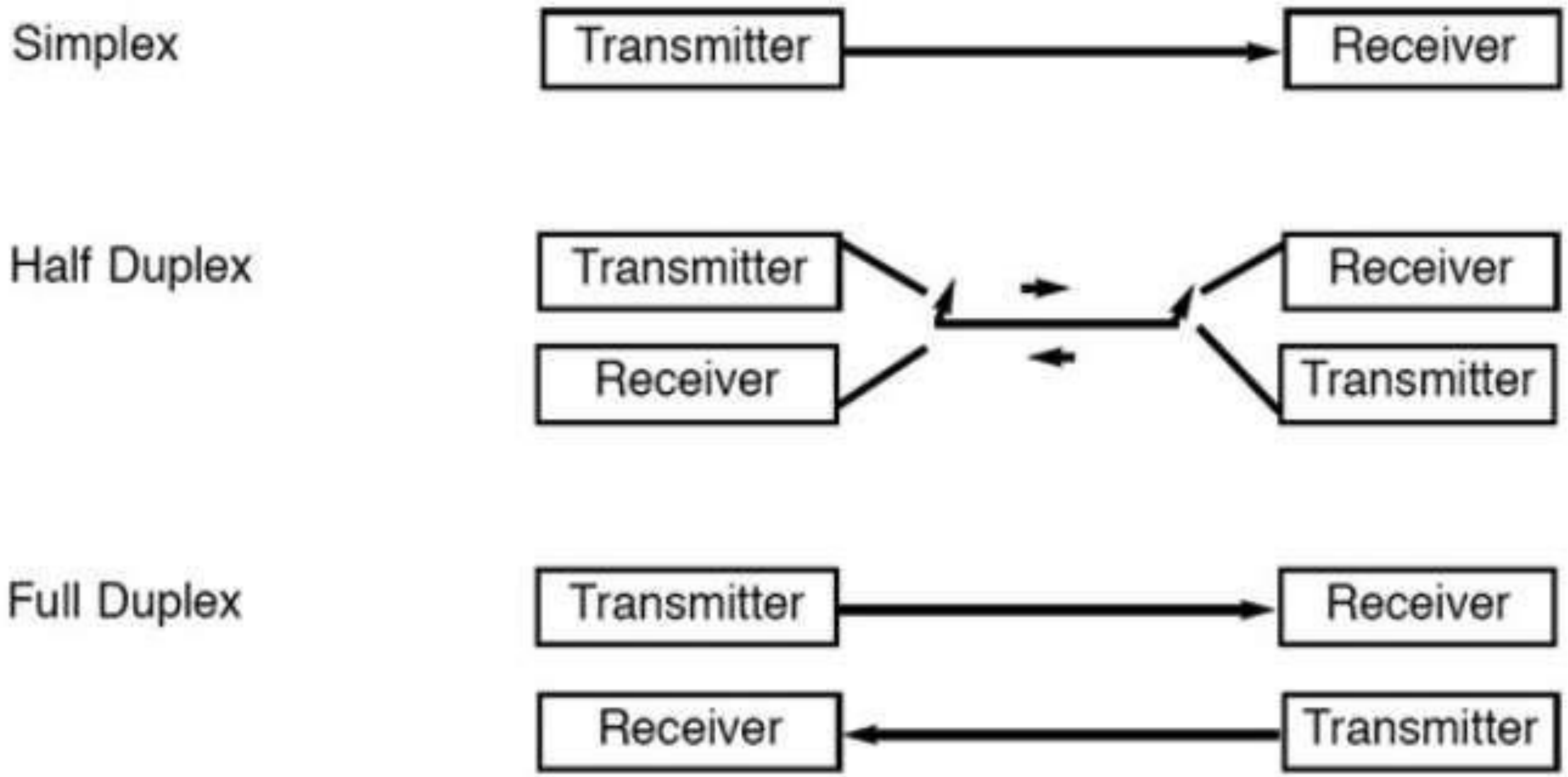
- There are **special IC's** made by many manufacturers for serial communications.
  - **UART** (universal asynchronous Receiver transmitter)
  - **USART** (universal synchronous-asynchronous Receiver-transmitter)

Serial communication using Max 232



Serial Communication

Classification based on direction of data transfer



Serial Communication

Classification based on direction of data transfer

- In simplex transmissions, the computer can only send data. There is only one wire.
- If the data can be transmitted and received, then it is a duplex transmission
- Duplex transmissions can be half or full duplex depending on whether or not the data transfer can be simultaneous
- If the communication is only one way at a time, it is half duplex
- If both sides can communicate at the same time, it is full duplex
    - ✓ Full duplex requires two wire conductors for the data lines (in addition to the signal ground)

Serial Communication

Classification based on Speed of data transfer

- Serial Communication can be
  - ✓ Asynchronous
  - ✓ Synchronous

Synchronous Communication

- Synchronous methods transfer a block of data (characters) at a time
- The events are referenced to a clock
- Example: SPI bus, I2C bus

Asynchronous Communication

- Asynchronous methods transfer a single byte at a time
- There is no clock. The bytes are separated by start and stop bits.
- Example: UART

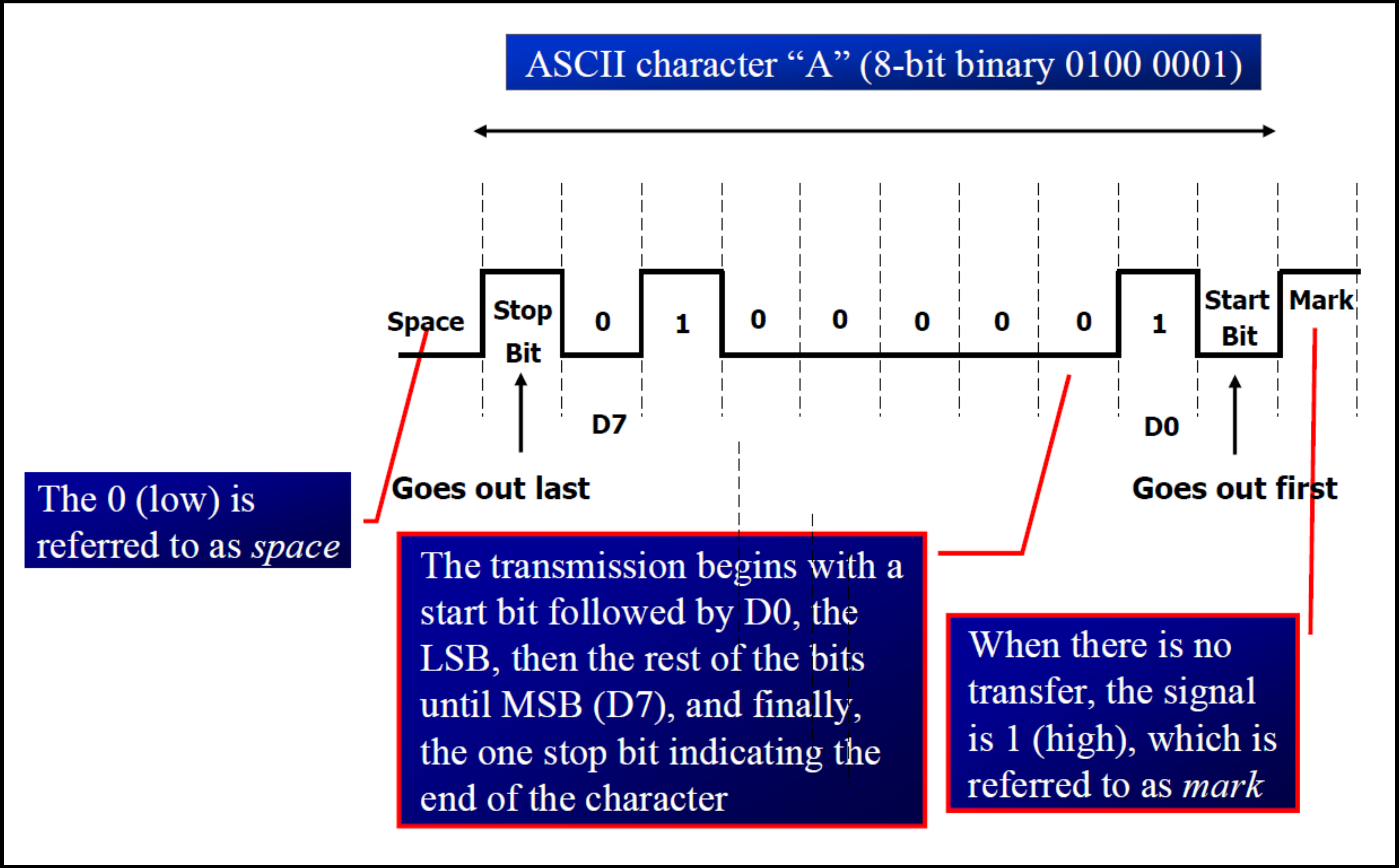
Asynchronous – Start & Stop Bit

- 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.

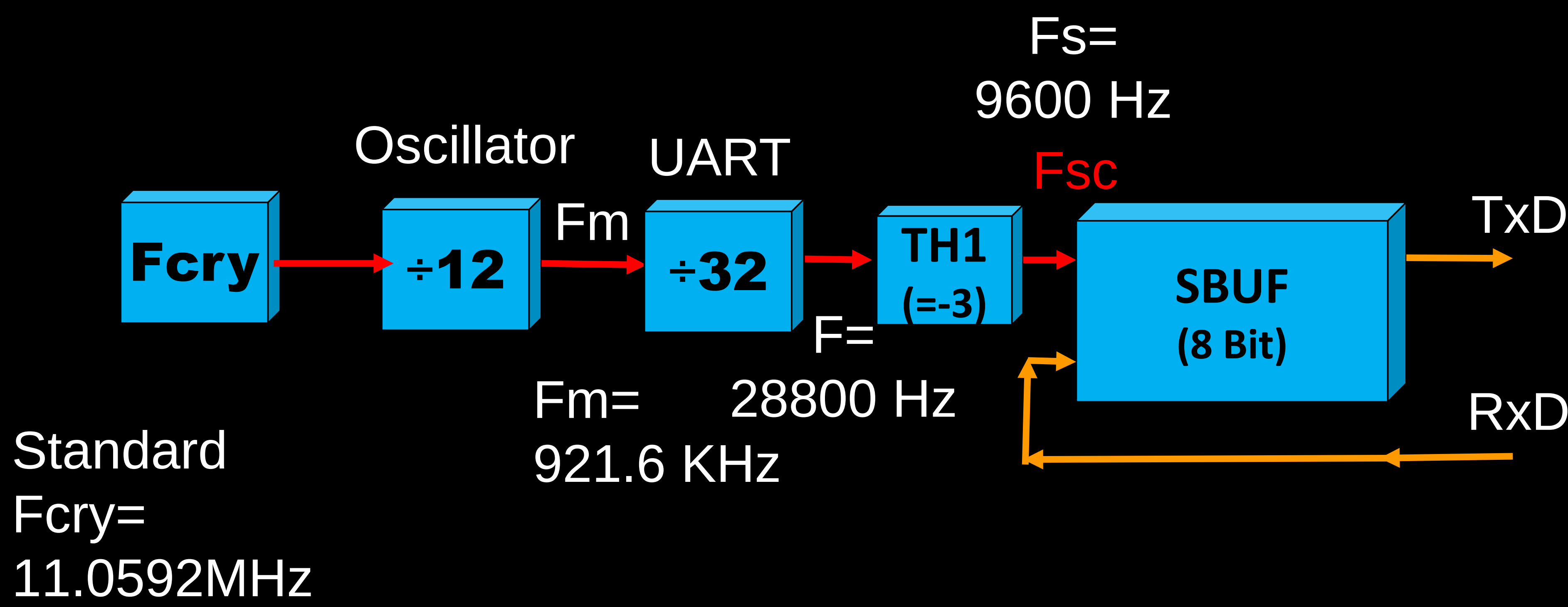
- **The start bit is always one bit, but the stop bit can be one or two bits**

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

Asynchronous – Start & Stop Bit



Serial Communication in 8051



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

SBUF Register

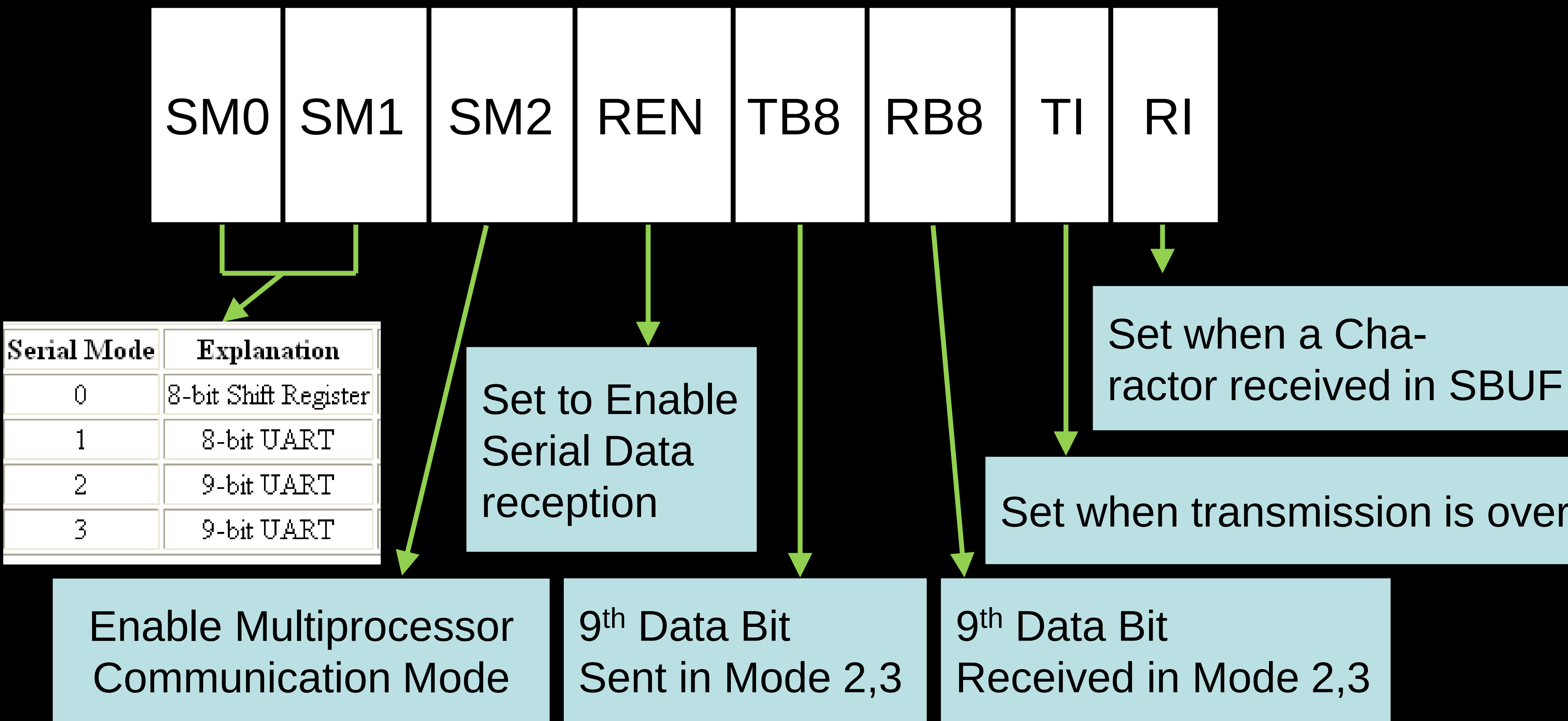
- **SBUF** is an **8-bit register** used solely for serial communication.
- For a byte data to be transferred via the **TxD line**, it must be placed in the **SBUF register**.
- The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD line.
- SBUF holds the byte of data when it is received by 8051 **RxD** line.
- When the bits are received serially via RxD, the **8051 deframes** it by eliminating the stop and start bits, making a byte out of the data received, and then placing it in SBUF.

SBUF Register

- **Sample Program:**

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

SCON Register



Serial Mode	Explanation
0	8-bit Shift Register
1	8-bit UART
2	9-bit UART
3	9-bit UART

8051 Serial Port – Mode 0

The Serial Port in Mode-0 has the following features:

1. Serial data **enters and exits through RXD**
2. **TXD** outputs the **clock**
3. 8 bits are transmitted / received
4. The baud rate is fixed at (1/12) of the oscillator frequency

8051 Serial Port – Mode 1

The Serial Port in Mode-1 has the following features:

1. Serial data **enters through RXD**
2. Serial data **exits through TXD**
3. On receive, the stop bit goes into RB8 in SCON
4. **10 bits** are transmitted / received
    1. *Start bit (0)*
    2. *Data bits (8)*
    3. *Stop Bit (1)*
5. Baud rate is determined by the Timer 1 over flow rate.

8051 Serial Port – Mode 2

The Serial Port in Mode-2 has the following features:

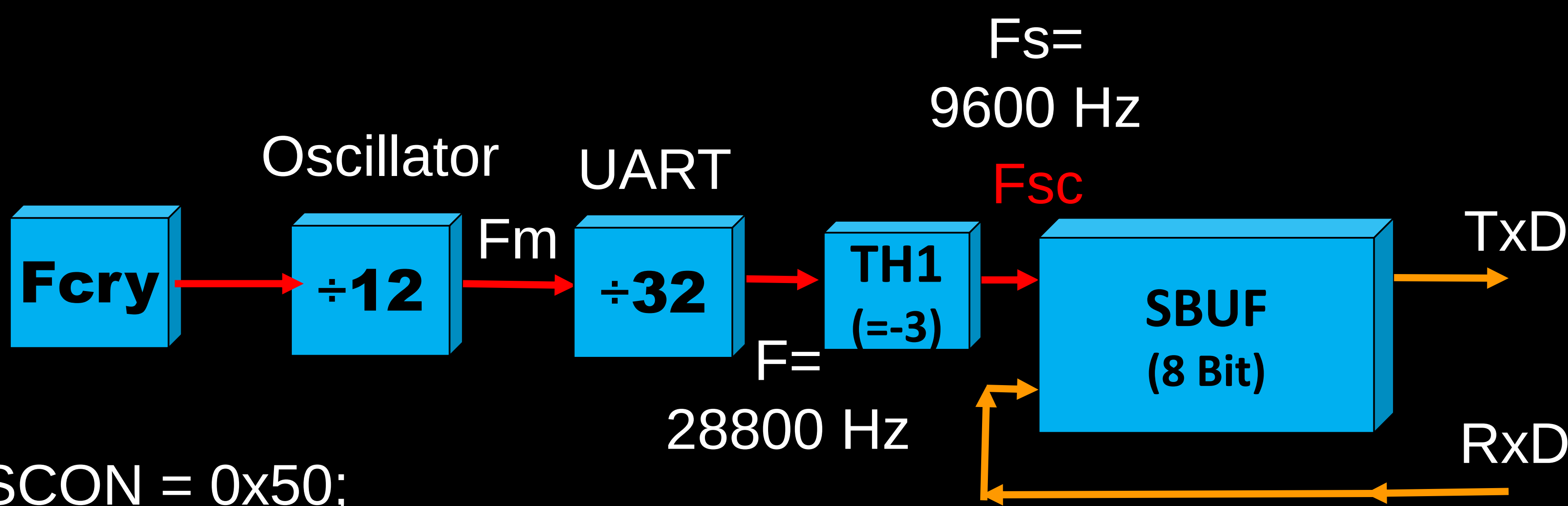
1. Serial data **enters through RXD**
2. Serial data **exits through TXD**
3. 9th data bit (**TB8**) can be assign value 0 or 1
4. On receive, the 9th data bit goes into **RB8** in SCON
5. **11 bits** are transmitted / received
    1. Start bit (0)
    2. Data bits (9)
    3. Stop Bit (1)
6. **Baud rate** is programmable

8051 Serial Port – Mode 3

The Serial Port in Mode-3 has the following features:

1. Serial data **enters through RXD**
2. Serial data **exits through TXD**
3. 9th data bit (**TB8**) can be assign value 0 or 1
4. On receive, the 9th data bit goes into **RB8** in SCON
5. **11 bits** are transmitted / received
    1. Start bit (0)
    2. Data bits (9)
    3. Stop Bit (1)
6. **Baud rate** is determined by Timer 1 overflow rate.

Programming Serial Data Transmission



1. SCON = 0x50;
2. TMOD = 0x20;
3. TH1=-3;
4. TR1=1;

5. SBUF=____
6. While(TI==0);
7. TI=0;

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

Programming Serial Data Transmission

1. The **SCON register** is loaded with the value **50H**, indicating serial mode 1, where an 8- bit data is framed with start and stop bits.
2. **TMOD register** is loaded with the value **20H**, indicating the use of timer 1 in mode 2 (8-bit auto-reload) **to set baud rate**.
3. The **TH1** is loaded with one of the values to set baud rate for serial data transfer.
4. **TR1** is set to 1 to start timer 1
5. The character byte to be transferred serially is written into **SBUF register.**
6. The **TI flag bit** is monitored with the use of instruction **JNB TI, xx** to see if the character has been transferred completely.
7. **TI** is cleared by **CLR TI** instruction
8. To transfer the next byte, **go to step 5**

Program…??

Programming Serial Data Transmission

1. Write a program to transmit “A” continuously to the PC at a baud rate of 9600 using serial communication in 8051 microcontroller.
2. **Write a program to transmit “Welcome to NFSU” on PC at a baud rate of 9600 using serial communication in 8051 microcontroller.**

Programming Serial Data Reception

1. **TMOD register** is loaded with the value **20H**, indicating the use of timer 1 in mode 2 (8-bit auto-reload) **to set baud rate.**

2. **TH1** is loaded to set baud rate

3. The **SCON register** is loaded with the value **50H**, indicating serial mode 1, where an 8- bit data is framed with start and stop bits.

4. **TR1** is set to 1 to start timer 1

6. The **RI flag bit** is monitored with the use of instruction **JNB RI, xx** to see if an entire character has been received yet

7. **When RI is raised, SBUF** has the byte, its contents are moved into a safe place.

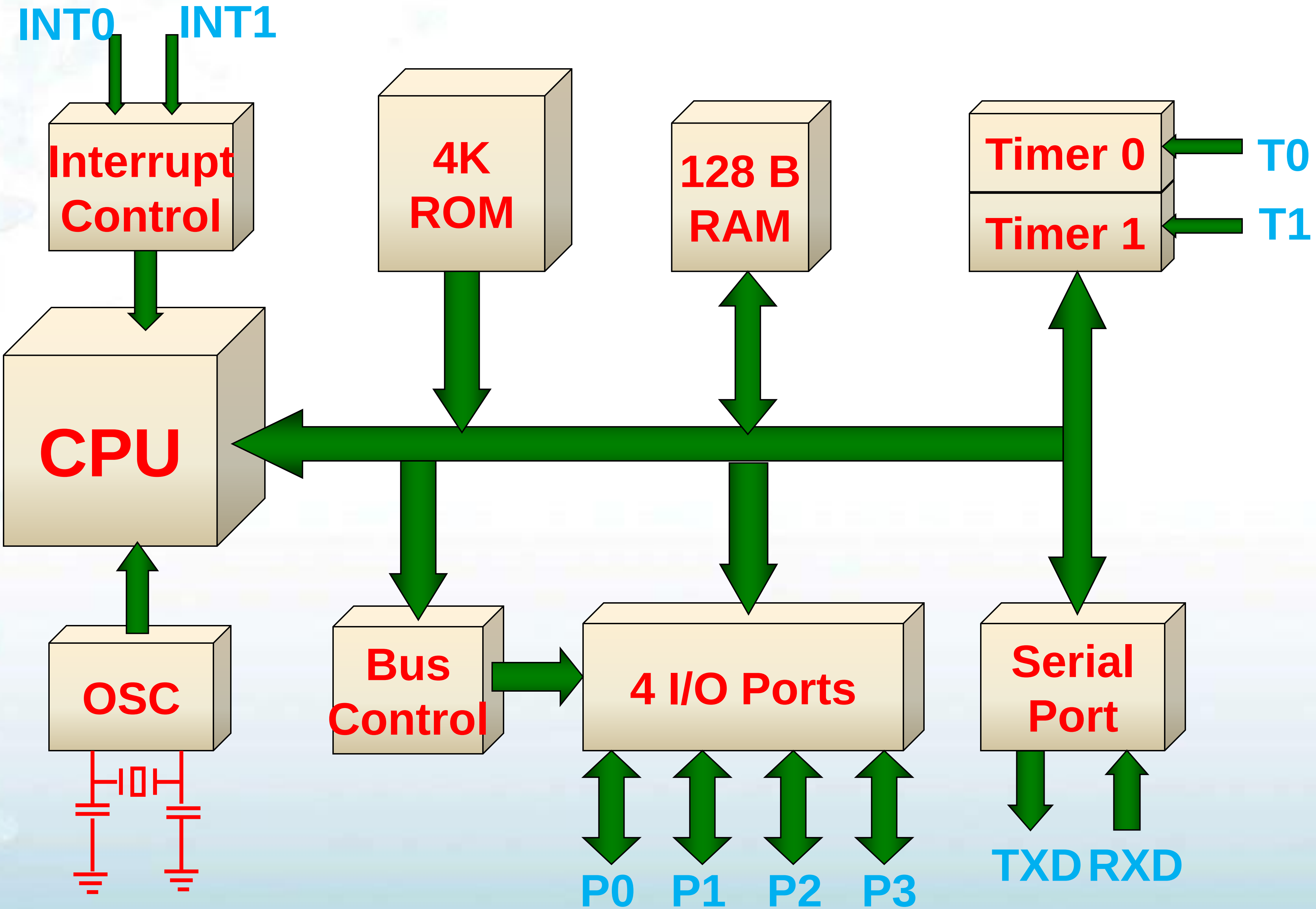
8. **RI** is cleared by **CLR RI** instruction

9. To receive the next character, **go to step 5.**

Program…??

8051 Interrupts

General Block Diagram of 8051



INTERRUPTS

- An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service

- A single microcontroller can serve several devices by two ways:

  1. **Interrupt**
  2. **Polling**

Interrupt Vs Polling

1. **Interrupts**
   - Whenever any device needs its service, the device notifies the microcontroller by sending it an **interrupt signal**.
   - Upon receiving an interrupt signal, the **microcontroller interrupts** whatever it is doing and serves the device.
   - The program which is associated with the interrupt is called the **interrupt service routine (ISR)** or interrupt handler.
2. **Polling**
   - The microcontroller **continuously monitors** the status of a given device.
   - When the **conditions** met, it performs the service.
   - After that, it moves on to monitor the **next device** until every one is serviced.

Interrupt Vs Polling

- The **polling method is not efficient**, since it wastes much of the microcontroller's time by polling devices that do not need service.

- The **advantage of interrupts** is that the microcontroller can serve many devices (not all at the same time).

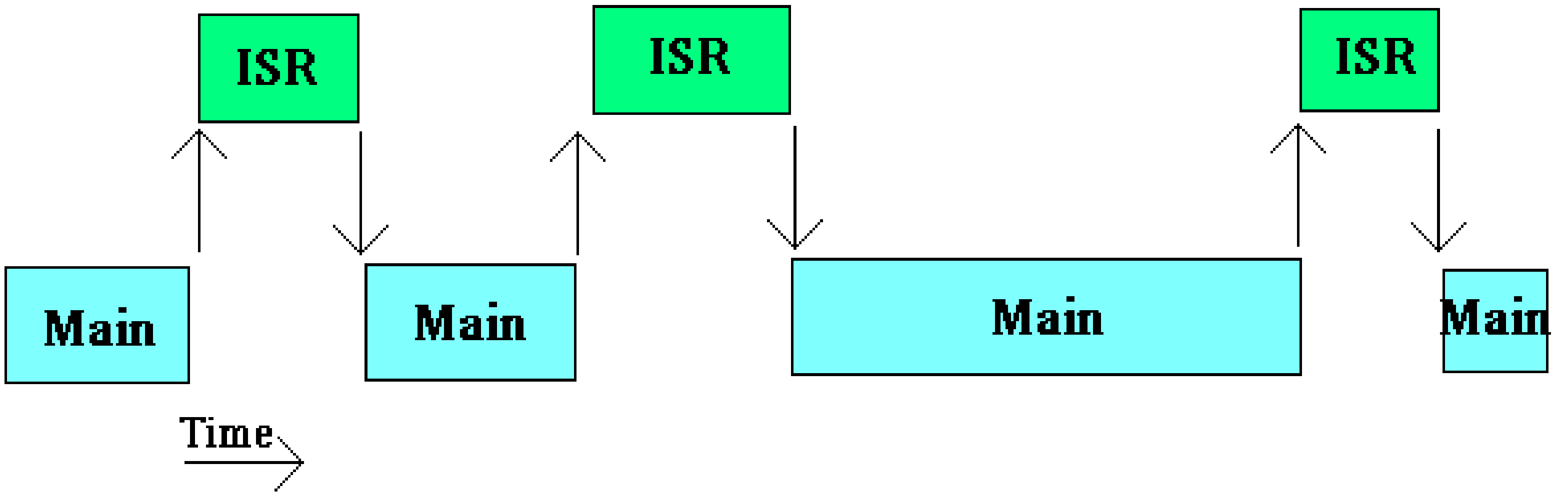
- Each devices can get the attention of the microcontroller based on the **assigned priority**.

- For the polling method, it is **not possible** to assign priority since it checks all devices in a round-robin fashion.

- The microcontroller can also **ignore (mask)** a device request for service in Interrupt.

Program execution without intrrupts :

Program execution with intrrupts :



ISR : Intrrupt Service Routin

Six Interrupts in 8051

Six interrupts are allocated as follows:

1. **Reset – power-up reset.**

2. **Two interrupts are set aside for the timers.**
   - one for timer 0 and one for timer 1

3. **Two interrupts are set aside for hardware external interrupts.**
   - P3.2 and P3.3 are for the external hardware interrupts INT0 (or EX1), and INT1 (or EX2)

4. **Serial communication has a single interrupt that belongs to both receive and transfer.**

What events can trigger Interrupts?

- We can configure the 8051 so that any of the following events will cause an interrupt:

  - Timer 0 Overflow.
  - Timer 1 Overflow.
  - Reception/Transmission of Serial Character.
  - External Event 0.
  - External Event 1.

- We can configure the 8051 so that when Timer 0 Overflows or when a character is sent/received, the appropriate interrupt handler routines are called.

8051 Interrupt Vectors

INTERRUPT VECTORS

When the original 8051 and 8031 were introduced, only 5 interrupts were provided.

Interrupt Number	Interrupt Vector Address	Description
0	0003h	EXTERNAL 0
1	000Bh	TIMER/COUNTER 0
2	0013h	EXTERNAL 1
3	001Bh	TIMER/COUNTER 1
4	0023h	SERIAL PORT

Enabling and Disabling an Interrupt

- Upon **reset**, all interrupts are **disabled (masked),** meaning that none will be responded to by the microcontroller if they are activated.

- The interrupts must be **enabled** by software in order for the microcontroller to respond to them.

- There is a register called **IE (interrupt enable)** that is responsible for enabling (unmasking) and disabling (masking) the interrupts.

Interrupt Enable (IE) Register

EA	—	--	ES	ET1	EX1	ET0	EX0

- EA : Global enable/disable.
- --- : Reserved for additional interrupt hardware.
- ES : Enable Serial port interrupt.
- ET1 : Enable Timer 1 control bit.
- EX1 : Enable External 1 interrupt.
- ET0 : Enable Timer 0 control bit.
- EX0 : Enable External 0 interrupt.

MOV IE,#08h
or
SETB ET1

Enabling and Disabling an Interrupt

- **Example:** Show the instructions to

(a) enable the serial interrupt, timer 0 interrupt, and external hardware interrupt 1 and

(b) disable (mask) the timer 0 interrupt, then

(c) Show how to disable all the interrupts with a single instruction.

- **Solution:**
  - **(a) MOV IE,#10010110B** ;enable serial, timer 0, EX1
    - Another way to perform the same manipulation is:
      - **SETB IE.7** ;EA=1, global enable
      - **SETB IE.4** ;enable serial interrupt
      - **SETB IE.1** ;enable Timer 0 interrupt
      - **SETB IE.2** ;enable EX1
  - **(b) CLR IE.1** ;mask (disable) timer 0 interrupt only
  - **(c) CLR IE.7** ;disable all interrupts

Interrupt Priority

- When the 8051 is powered up, the priorities are assigned according to the following.

- In reality, the priority scheme is nothing but an internal polling sequence in which the 8051 polls the interrupts in the sequence listed and responds accordingly.

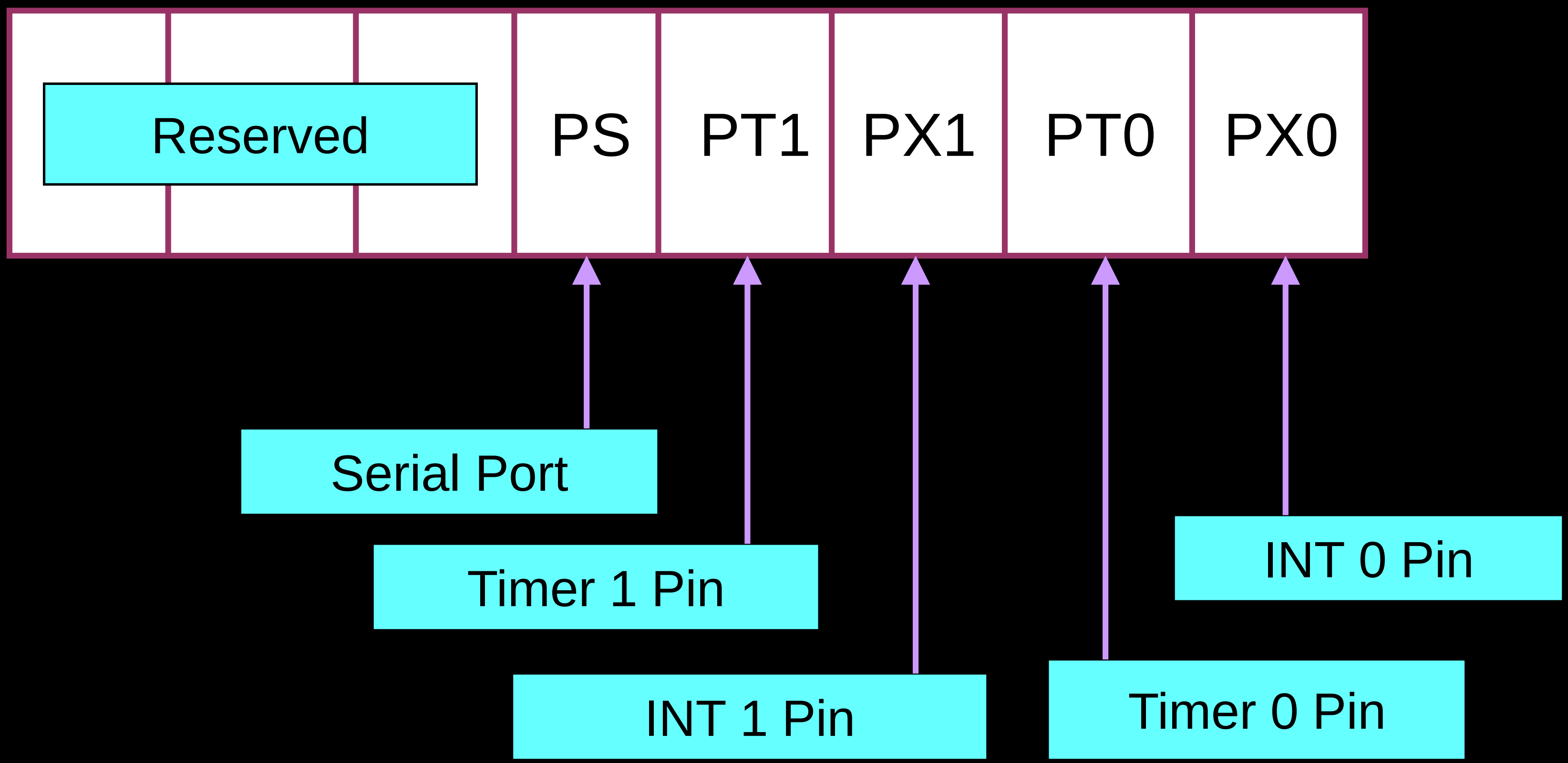
Interrupt vector table

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

Interrupt Priority

- **We can alter** the sequence of interrupt priority by assigning a higher priority to any one of the interrupts by programming a register called **IP (interrupt priority).**
- To give a higher priority to any of the interrupts, we make the **corresponding bit in the IP register high.**

Interrupt Priority (IP) Register



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

Example 11-14

Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0, while simultaneously creating a square wave of 200 μs period on pin P2.5. Use Timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

- Tr = 100 μs
- Count = Tr/T = 100 μs / 1.085 μs = 92
- Tx = n –count = 256 -92 = 164 = A4h

Example 11-14

Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0, while simultaneously creating a square wave of 200 μs period on pin P2.5. Use Timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

```
include <reg51.h>

sbit x=P1^7;
sbit y=P1^0;
sbit sw=P2^5;
void main()
{
        x=1;
        TMOD=0X02;
        TH0=0XA4;
        TR0=1;
        IE=0X82;
                while(1)
                {
                        y=x;
                }
}
```

```
void timer0(void) interrupt 1()
{
        sw=~sw;
}
```

Conclusion

- Speed with command is the secret of success.
- Be proactive and adaptive.
- Technological integration is the real education for any industry.
- Time & opportunity do not wait for anyone.