MODULE-4
Microcontroller 8051 Peripherals

V.PRAKASH
Asst. Professor(Sr.), SENSE,
VIT Chennai

MODULE-4

Microcontroller 8051 Peripherals

I/O Ports, Timers-Counters, Serial Communication and Interrupts.

I/O PORTS

I/O PORTS

- 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as inputs or outputs.
- Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.
- Except port 1 all other ports are used for dual purpose
- All ports are bidirectional and they are constructed with a D type output latch.
- Pin configuration, i.e. whether it is to be configured as an input (1) or an output (0), depends on its logic state.
- All the ports upon RESET are configured as input, ready to be used as input ports.

I/O PORTS

- Port 0
  - Port 0 occupies a total of 8 pins (pins 32-39) and it can be used for input or output.
  - To use the pins of port 0, each pin must be connected externally to a 10k ohm pull-up resistor. This is due to the fact that P0 is an open drain, unlike P1, P2, and P3.
  - Dual role of port-0: Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data. If ALE=1, then it is address or else data.
- Port 1
  - Port 1 occupies a total of 8 pins (pins 1-8) and it can be used for input or output.
  - In contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally.
  - Unlike other ports, port-1 doesn't used for dual purpose.

I/O PORTS

- Port 2
  - Port 2 occupies a total of 8 pins (pins 21- 28) and it can be used as input or output.
  - Just like P1, P2 does not need any pull-up resistors since it already has pull-up resistors internally.
  - Port 2 is also designated as A8 –A15, indicating its dual function and Port 0 provides the lower 8 bits via A0 –A7.
- Port 3
  - Port 3 occupies a total of 8 pins, (pins 10-17) and it can be used as input or output.
  - P3 does not need any pull-up resistors, the same as P1 and P2 did not.
  - Port 3 has the additional function of providing some extremely important signals such as interrupts.

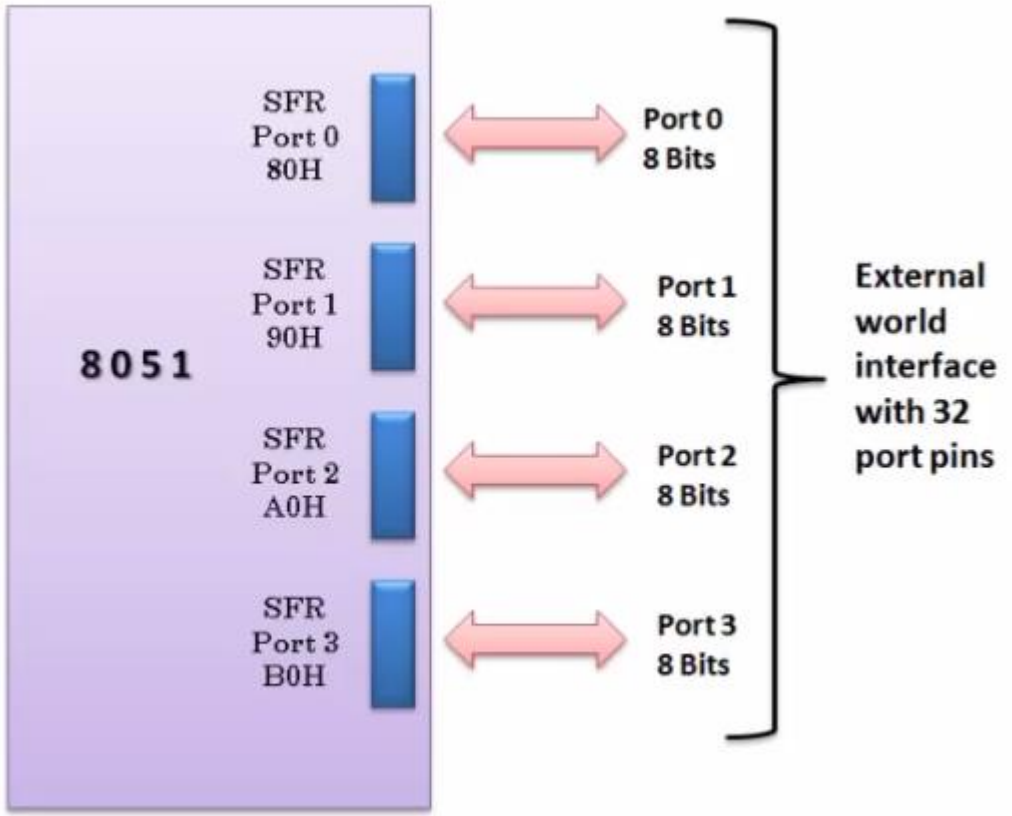
I/O PORTS

- I/O Pin's Current Limitations:
  - When configured as outputs (logic 0), single port pins can receive a current of 10mA.
  - If all 8 bits of a port are active, a total current must be limited to 15mA (port P0: 26mA).
  - If all ports (32 bits) are active, total maximum current must be limited to 71mA.
  - When these pins are configured as inputs (logic 1), built-in pull-up resistors provide very weak current, but strong enough to activate up to 4 TTL inputs of LS series.

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

I/O PORTS

8051 I/O PORTS & SFR BIT AND BYTE ADDRESS LOCATIONS



PORT 0 BITS	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
BIT ADDRESS	87H	86H	85H	84H	83H	82H	81H	80H

PORT 1 BITS	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
BIT ADDRESS	97H	96H	95H	94H	93H	92H	91H	90H

PORT 2 BITS	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
BIT ADDRESS	A7H	A6H	A5H	A4H	A3H	A2H	A1H	A0H

PORT 3 BITS	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
BIT ADDRESS	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H

I/O PORTS

EXAMPLE-1

Write an 8051 assembly code to configure port-1 as output port and send out an alternating value 55H and AAH continuously with a delay between each data out.

```
        MOV P1, #00H
BACK:   MOV A, #55H
        MOV P1,A
        ACALL DELAY
        MOV A, #AAH
        MOV P1,A
        ACALL DELAY
        SJMP BACK

DELAY:  MOV R3, #250
HERE:   NOP
        NOP
        NOP
        NOP
        DJNZ R3, HERE
        RET
```

I/O PORTS

EXAMPLE-2

Write an 8051 assembly code to generate a square wave of 50% duty cycle on bit 0 of port 1.

Solution:

The 50% duty cycle means that HIGH (ON state) and LOW (OFF state) of the pulse should have the same length. So, we have to toggle P1.0 with a time delay in between each state.

```
        CLR P1.0
BACK:   SETB P1.0
        ACALL DELAY
        CLR P1.0
        ACALL DELAY
        SJMP BACK
```

```
DELAY:  MOV R3, #250
HERE:   NOP
        NOP
        NOP
        NOP
        DJNZ R3, HERE
        RET
```

Exercise: Write an 8051 assembly code to generate a square wave of 25% duty cycle on bit 0 of port 1.

I/O PORTS

EXAMPLE-3

Assume that bit P2.3 is an input and represents the condition of an oven. If it goes high, it means that the oven is hot. Monitor the bit continuously. Whenever it goes high, send a high-to-low pulse to port P1.5 to turn on a buzzer. Write an 8051 assembly code for this condition.

```
        SETB P2.3          ; Configure P2.3 as input port pin
HERE:   JNB P2.3,HERE      ;keep monitoring for high
        SETB P1.5          ;set bit P1.5=1
        CLR P1.5           ;make high-to-low
        SJMP HERE          ;keep repeating
```

Exercise: Write an 8051 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.

I/O PORTS

EXAMPLE-4

A switch is connected to pin P1.7. Write an 8051 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

```
        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
```

Exercise: A switch is connected to pin P1.0 and an LED to pin P2.7. Write an 8051program to get the status of the switch and send it to the LED

TIMERS / COUNTERS

TIMERS / COUNTERS

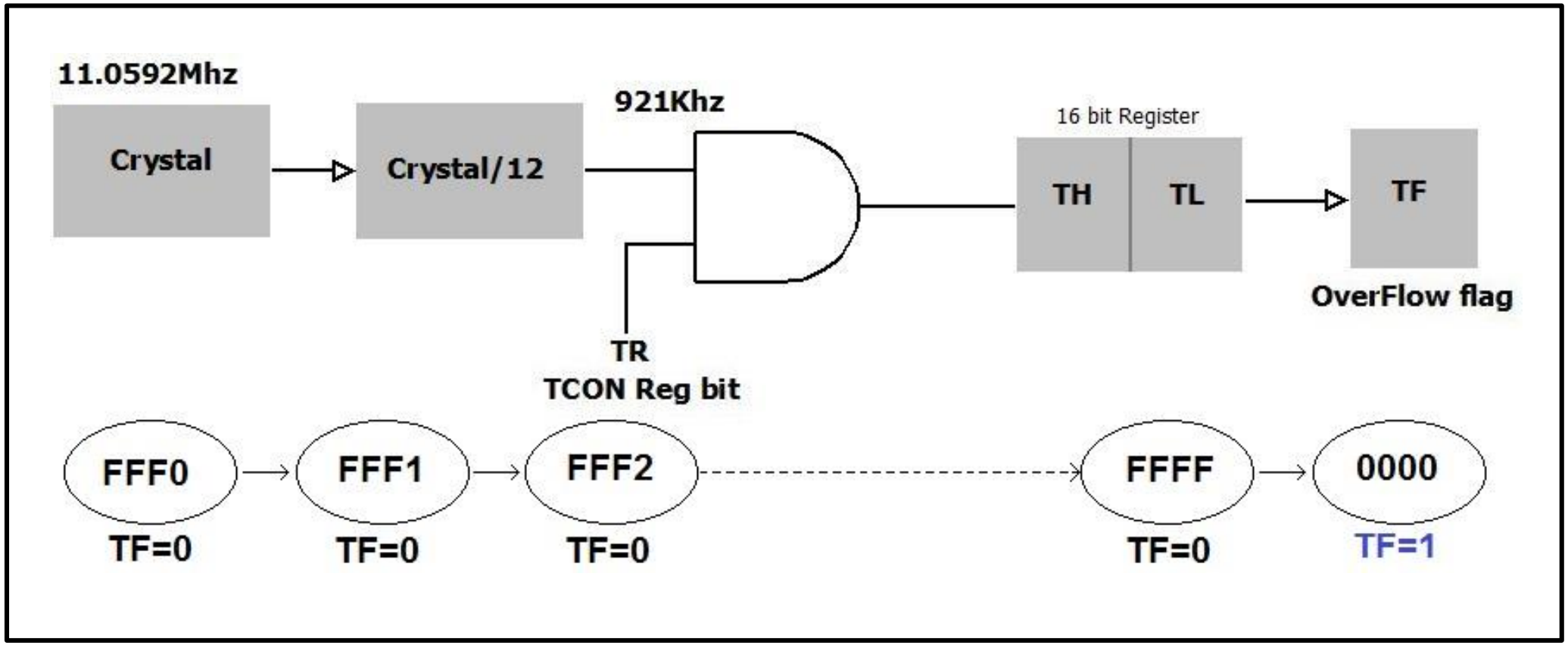
TIMERS

- 8051 has two timers/counters (Timer 0 and Timer 1), both are 16 bits counter. They can be used either as
  - ✓ Timers to generate a time delay
  - ✓ To generate a waveform with specific frequency
  - ✓ To generate baud rate signal for serial communication
  - ✓ Event counters to count events happening outside the microcontroller

- Register related to work with 8051 timers are:
  1. TH & TL Timer/counter register– To hold the value for generating time delay
  2. TMOD Register - to define mode of timer operation
  3. TCON Register – To control the timer operation

TIMERS / COUNTERS

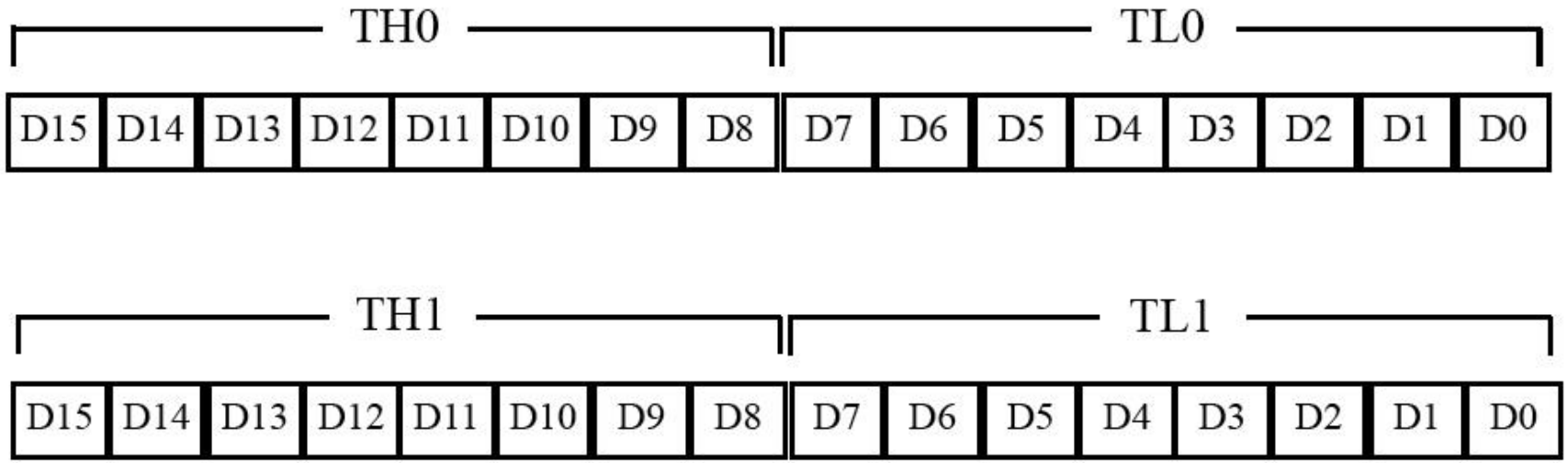
TIMERS/COUNTER REGISTERS



TIMERS / COUNTERS

TIMERS/COUNTER REGISTERS

- 8051 has two timers/counters registers (Timer 0 and Timer 1), each 16-bits wide and accessed as two separate registers of low byte and high byte
  - ✓ The low byte register is called TL0/TL1 and
  - ✓ The high byte register is called TH0/TH1



TIMERS / COUNTERS

TIMERS/COUNTER REGISTERS

- Steps to calculate values to be loaded into the TL and TH registers
  1. Divide the desired time delay by 1.085us (if operating frequency is 11.0592 MHz)
  2. Perform 65536 – n, where "n" is the decimal value got in Step1 (if 16-bit counter mode is selected)
  3. Convert the result of Step2 to hex value and represent in four digits (ex: xxyy)
  4. If "xxyy" is the hex value to be loaded into the timer's register, then set TH = xx and TL = yy

- Example: 500us time delay

  Step1: 500us/1.085us = 461pulses

  Step2: P=65536-461=65075

  Step3: 65074 converted by hexa decimal =FE33

  Step4: TH1=0xFE; TL1=0x33;

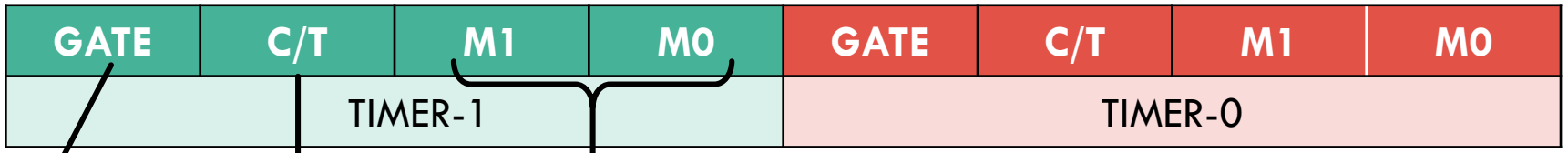
TIMERS / COUNTERS

TMOD REGISTER

- Both timers 0 and 1 use the same register, called TMOD (timer mode), to set the various timer operation modes
- TMOD is a 8-bit register:
  - The lower 4 bits are for Timer 0
  - The upper 4 bits are for Timer 1
- Timers of 8051 do starting and stopping by either software or hardware control
- Software: First set GATE=0, then start and stop of the timer are controlled by the TR(timer start) bits TR0 and TR1. The SETB starts it, and it is stopped by the CLR instruction.
- Hardware: The starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register.

TIMERS / COUNTERS

TMOD REGISTER



Gating Control:

- If GATE=1 – Timer/Counter is enable only while INTx pin is HIGH and the TRx control pin is SET
- If GATE=0, the timer is enabled whenever the TRx control bit is set

Counter/Timer Selction:

- C/T=1 – Act as a counter and receive external input signal from P3.5 pin for T1, (P3.4 for T0)
- C/T=0 – Act as a timer and receive input signal from internal system clock

M1	M0	Operating Mode
0	0	13-bit Timer Mode (Mode-0): 8-bit Timer/Counter THx with TLx as 5-bit prescaler
0	1	16-bit Timer Mode (Mode-1): 16-bit Timer/Counter THx and TLx are cascaded, no prescaler
1	0	8-bit Auto-reload Mode (Mode-2): 8-bit auto reload Timer/Counter; THx holds the value which is to be reloaded when TLx overflows each time
1	1	Split-timer Mode (Mode-3)

*Where x represent 0 for Timer0 and 1 for Timer 1

TIMERS / COUNTERS

TCON REGISTER

- TFx: Timerx Overflow Flag
  - TFx =1 means Timerx overflow occurred (i.e. Timerx goes to its max and roll over back to zero).
  - TFx =1 means Timerx overflow not occurred.

- TRx: Timerx Run Control Bit
  - TRx =1 means Timerx start.
  - TRx =0 means Timerx stop.

- ITx: External Interruptx Trigger Type Select Bit
  - ITx=1 means Interrupt occurs on falling edge at INTx pin.
  - ITx=0 means Interrupt occur on a low level at the INTx pin.

- IEx: External Interruptx Edge Flag
  - IEx=1 means External interruptx occurred.
  - IEx=0 means External interruptx Processed.

*Where x represent 0 for Timer0 and 1 for Timer 1

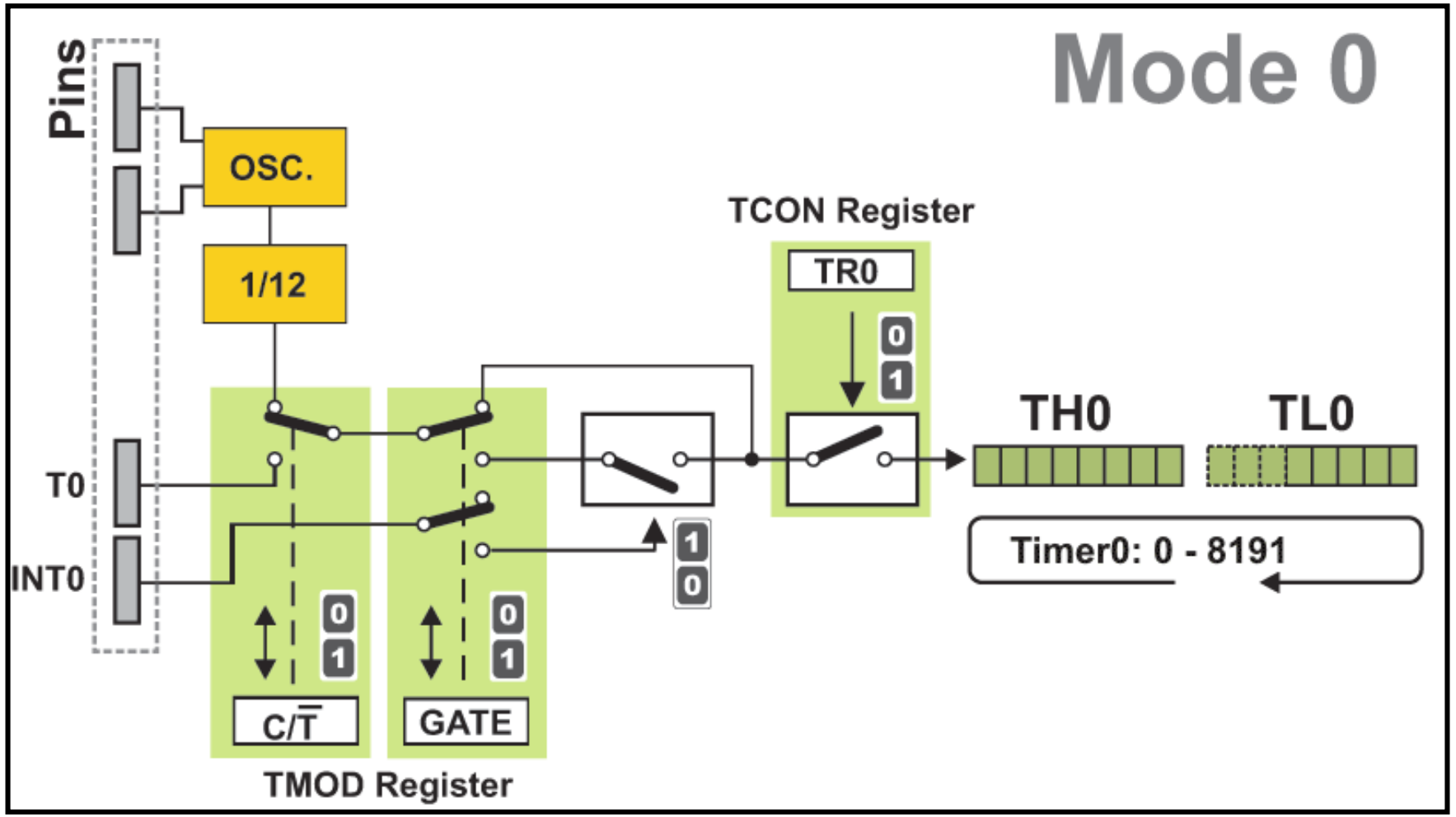
TIMERS / COUNTERS

TIMER 0 MODE 0 (13-BIT TIMER)

- This is one of the rarities being kept only for the purpose of compatibility with the previous versions of microcontrollers.
- This mode configures timer 0 as a 13-bit timer which consists of all 8-bits of TH0 and the lower 5-bits of TL0.
- How does it operate? Each coming pulse causes the lower register bits to change their states. After receiving 32 pulses, this register is loaded and automatically cleared, while the higher byte (TH0) is incremented by 1.
- This process is repeated until registers count up 8192 pulses. After that, both registers are cleared and counting starts from 0

TIMERS / COUNTERS

TIMER 0 MODE 0 (13-BIT TIMER)



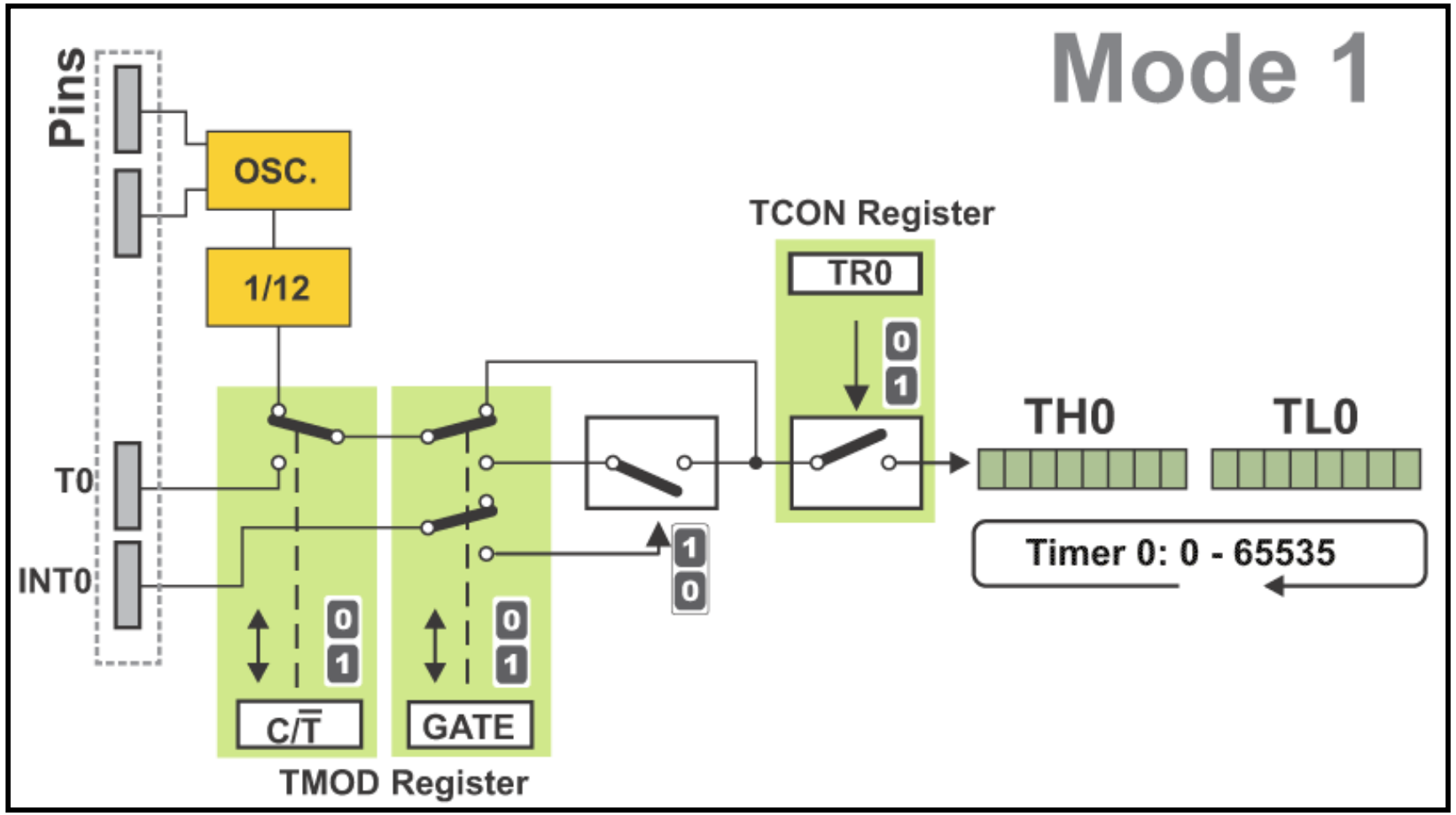
TIMERS / COUNTERS

TIMER 0 MODE 1 (16-BIT TIMER)

- It is a 16-bit timer; therefore it allows values from 0000 to FFFFH to be loaded into the timer's registers TL and TH.
- After TH and TL are loaded with a 16-bit initial value, the timer must be started.
- After the timer is started. It starts count up until it reaches its limit of FFFFH.
- When it rolls over from FFFF to 0000H, it sets high a flag bit called TF (timer flag).
- This is one of the most commonly used modes.

TIMERS / COUNTERS

TIMER 0 MODE 1 (16-BIT TIMER)



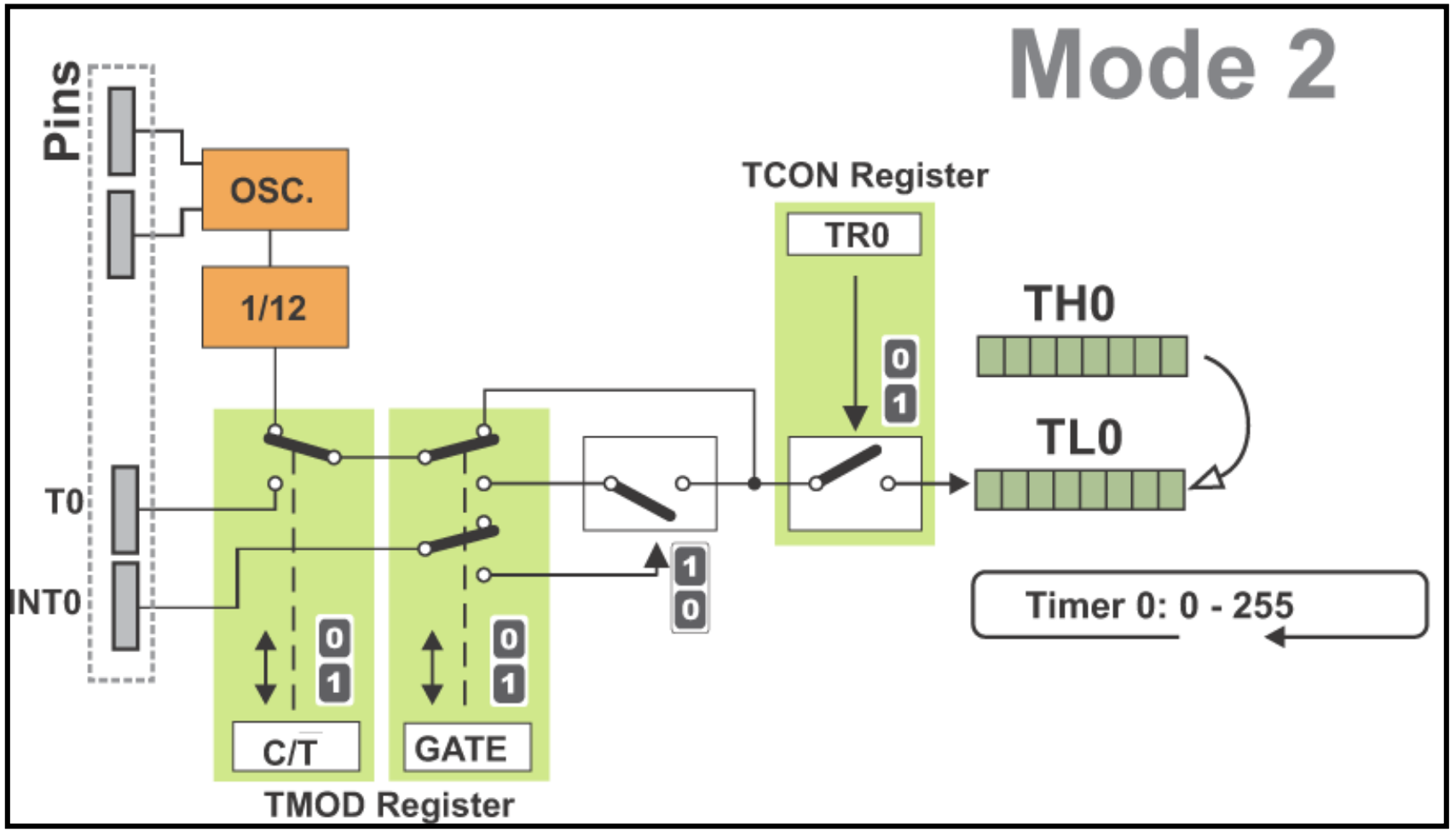
TIMERS / COUNTERS

TIMER 0 MODE 2 (8-BIT AUTO-RELOAD TIMER)

- When a timer is in mode 2, THx holds the "reload value" and TLx operates as a timer.
- Thus, TLx starts counting up. When TLx reaches 255 and is subsequently incremented, instead of resetting to 0 (as in the case of modes 0 and 1), it will be reset to the value stored in THx.
- The auto-reload mode is very commonly used for establishing a baud rate.
- For example, suppose it is necessary to constantly count up 55 pulses generated by the clock.
- In order to register each 55th pulse, the best solution is to write the number 200 to the TH0 register and configure the timer to operate in mode 2.

TIMERS / COUNTERS

TIMER 0 MODE 2 (8-BIT AUTO-RELOAD TIMER)



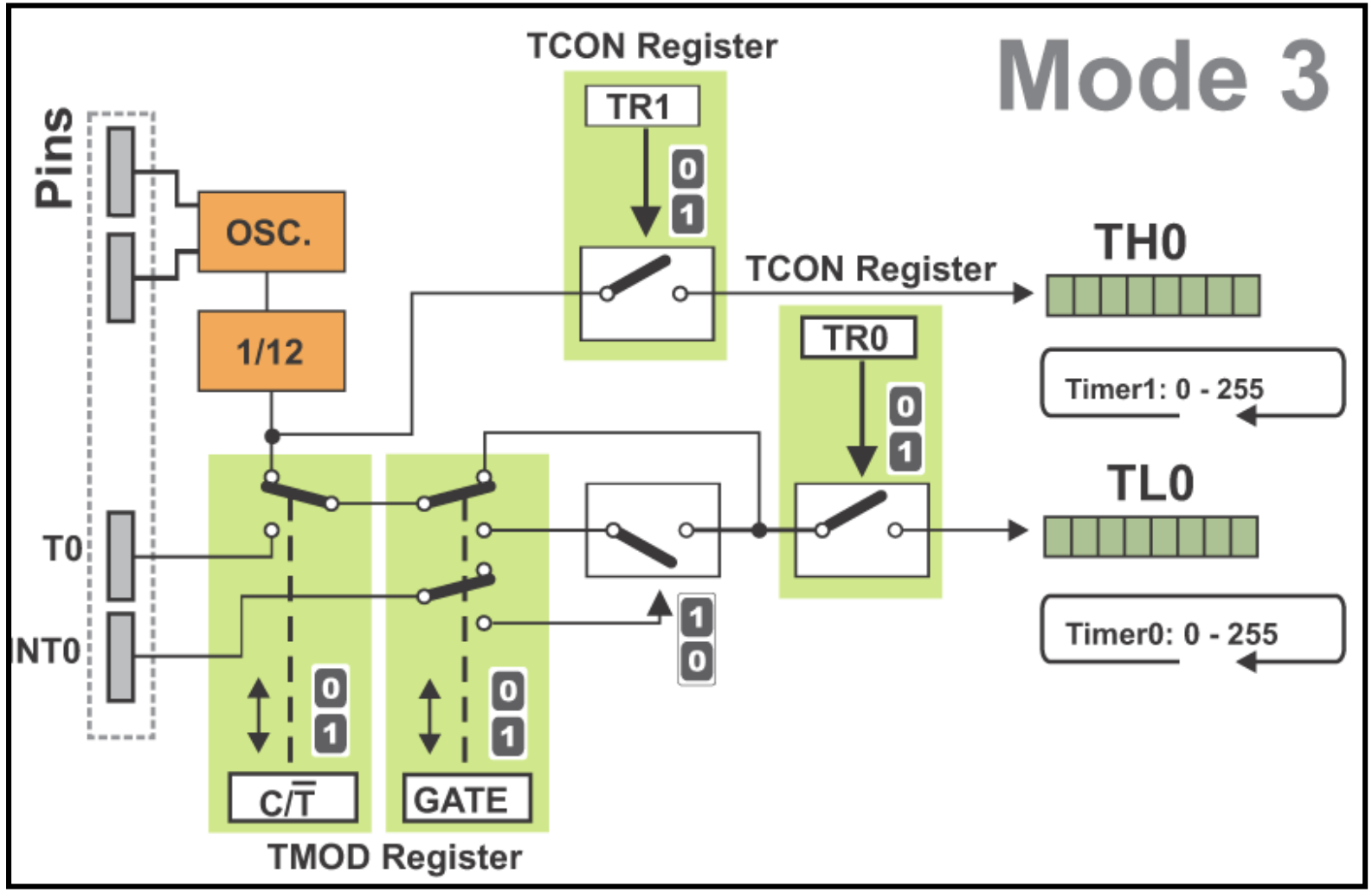
TIMERS / COUNTERS

TIMER 0 MODE 3 (SPLIT TIMER)

- This mode is provided for applications requiring an additional 8-bit timer or counter.
- Mode 3 configures timer 0 so that registers TL0 and TH0 operate as separate 8-bit timers.
- In other words, the 16-bit timer 0 consisting of two registers TH0 and TL0 is split into two independent 8-bit timers.
- The TL0 timer turns into timer 0, while the TH0 timer turns into timer 1.
- In addition, all the control bits of 16-bit Timer 1 (consisting of the TH1 and TL1 register), now control the 8-bit Timer 1.
- While Timer 0 is in split mode, you may not start or stop the real timer 1 since the bits that do that are now linked to TH0.

TIMERS / COUNTERS

TIMER 0 MODE 3 (SPLIT TIMER)



TIMERS / COUNTERS

STEPS TO PROGRAM 0 MODE 1 (16-BIT TIMER)

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) and timer mode (0 or 1) is selected. MOV TMOD,#01H.

2. Load registers TL and TH with initial count value. MOV TH0,#FFH; MOV TL0,#FCH.

3. Start the timer using SETB TRx. SETB TR0.

4. Keep monitoring the timer flag (TF). AGAIN: JNB TF0, AGAIN.

5. Stop the timer using CLR TRx. CLR TR0.

6. Clear the TF flag for the next round. CLR TF0.

7. Go back to Step 2 to load TH and TL again. SJMP STEP2.

TIMERS / COUNTERS

LARGEST POSSIBLE DELAY USING TIMERS

Calculate TL and TH to get the largest time delay possible. Find the delay in ms. In your calculation, exclude the overhead due to the instructions in the loop

Solution:

- To get the largest delay we make TL and TH both 0. This will count up from 0000 to FFFFH and then roll over to zero
- Making TH and TL both zero means that the timer will count from 0000 to FFFF, and then roll over to raise the TF flag.
- As a result, it goes through a total Of 65536 states. Therefore, we have delay =(65536 - 0) ×1.085 us = 71.1065ms.

TIMERS / COUNTERS

EXAMPLE-1

Write an 8051 assembly language program to blink an LED connected in P2.7 with a time delay of 5ms by using timers. Assume that XTAL = 11.0592 MHz.

- Since XTAL = 11.0592 MHz, the counter counts up every 1.085 us.
- We must identify how many times 1.085 us intervals should be repeated to achieve 5 ms delay.
- To get that, perform 5 ms / 1.085 us = 4608 clocks.
- To Achieve that we need to load into TL and TH the value 65536 – 4608 = 60928.
- Convert 60928 into HEX i.e EE00. Therefore, we have TH = EE and TL = 00.

TIMERS / COUNTERS

EXAMPLE-1

```
        CLR P2.7          ;Clear P2.7
        MOV TMOD,#01      ;Timer 0, 16-bitmode
HERE:   MOV TL0,#0        ;TL0=0, the low byte
        MOV TH0,#0EEH     ;TH0=EE, the high byte
        CPL P2.7          ;SET high P2.7
        SETB TR0          ;Start timer 0
AGAIN:  JNB TF0,AGAIN     ;Monitor timer flag 0
        CLR TR0           ;Stop the timer 0
        CLR TF0           ;Clear timer 0 flag
        SJMP HERE
```

Exercise: Assuming XTAL = 22MHz write a program to generate a square pulse of 200ms period with 75% duty cycle on pin P2.4. Use timer 1 mode.

EXAMPLE-2

Generate a square wave with TON of 3ms and TOFF of 10ms on all pins of port 0. Assume an XTAL of 22MHz.

For 22MHz , one timer cycle time is

- 22MHz / 12 = 1.83MHz
- T = 1/1.83M = 0.546 us

For OFF time calculation:

- 10ms/0.546 us = 18315 cycles
- 65536-18315 =47221 = B875H

For ON time calculation:

- 3ms/0.546us = 5494 cycles
- 65536 – 5494 = 60042=EA8AH

TIMERS / COUNTERS

EXAMPLE-2

```
        MOV TMOD, #01H
BACK:   MOV TL0, #75H
        MOV TH0, #0B8H
        MOV P0, #00H
        ACALL DELAY
        MOV TL0, #8AH
        MOV TH0, #0EAH
        MOV P0, #0FFH
        ACALL DELAY
        SJMP BACK
```

```
        ORG 300H
DELAY:  SETB TR0
AGAIN:  JNB TF0, AGAIN
        CLR TR0
        CLR TF0
        RET
```

Exercise: Generate a square wave with TON of 5ms and TOFF of 2ms on all pins of port 0. Assume an XTAL of 11.0592 MHz.

EXAMPLE-3

Assuming XTAL = 22MHz write a program to generate a square pulse of frequency 100kHz on port pin P1.2 using timer-1 mode 2.

For 22MHz , one timer cycle time is

- 22MHz / 12 = 1.83MHz
- T = 1/1.83M = 0.546 μs

For 100kHz square wave,

- T=1/f = 0.01ms = 10μS
- TON = TOFF = 10uS/2 = 5μS
- 5 μs/ 0.546 μs = 9 cycles
- 256 – 9 = 247 = F7H

```
MOV TMOD, #20H
MOV TH1, #0F7H
SETB TR1
BACK: JNB TF1, BACK
CPL P1.2
CLR TF1
SJMP BACK
```

Exercise: Assume that XTAL = 11.0592 MHz, write a 8051 program to generate a square wave of 2 kHz frequency on pin P1.7

TIMERS / COUNTERS

COUNTERS

- Timers can also be used as counters counting events happening outside the 8051
  - In counter, it is a pulse outside of the 8051 that increments the TH, TL registers
  - TMOD and TH, TL registers are the same as for the timer discussed previously
- Programming the timer in the last section also applies to programming it as a counter except the source of the frequency
- The C/T bit in the TMOD registers decides the source of the clock for the timer
- When C/T = 1, the timer is used as a counter and gets its pulses from outside the 8051
- The counter counts up as pulses are fed from pins 14 and 15, these pins are called T0 (timer 0 input) and T1 (timer 1 input).

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

TIMERS / COUNTERS

EXAMPLE-4

Assuming that clock pulses are fed into pin T1(P3.5), write a 8051 assembly program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P1, which connects to 8 LEDs.

```
        ORG 0000H
        SETB P3.5           ; make T1 input
        MOV P1,#00H         ; Make Port 1 as output port
        MOV TMOD, #60H      ; counter 1, mode 2;C/T=1 external pulses
        MOV TH1, #0         ; clear TH1
AGAIN:  SETB TR1            ; start the counter
BACK:   MOV A, TL1          ; get copy of TL
        MOV P1, A           ; display it on port 2
        JNB TF1, BACK       ; keep doing, if TF=0
        CLR TR1             ; Stop the counter 1
        CLR TF1             ; make TF=0
        SJMP AGAIN          ; keep doing it
```

TIMERS / COUNTERS

EXAMPLE-5

Write an 8051 assembly language program to implement a counter for counting pulses of an input signals. Assume the crystal frequency as 22 MHz. Configure TIMER 1 to generate a clock pulse for every one seconds at P3.5 and TIMER 0 as a counter which receives input pulses at P3.4 from P3.5. Display final count values in port P1 (TL0) & P2 (TH0).

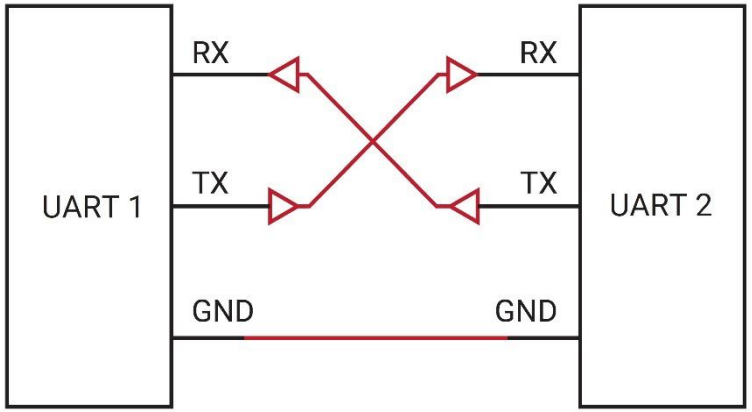
```
        ORG 000H
REPEAT: MOV TMOD, #15H
        SETB P3.4
        CLR P3.5
        MOV TL0, #00
        MOV TH0, #00
        SETB TR0
        MOV R0,#28
AGAIN:  MOV TL1,#00
        MOV TH1, #00
        SETB TR1
        ……
        ……
BACK:   JNB TF1, BACK
        CLR TF1
        CLR TR1
        DJNZ R0, AGAIN
        MOV A, TL0
        MOV P1,A
        MOV A, TH0
        MOV P2,A
        SJMP REPEAT
        END
```

SERIAL COMMUNICATION

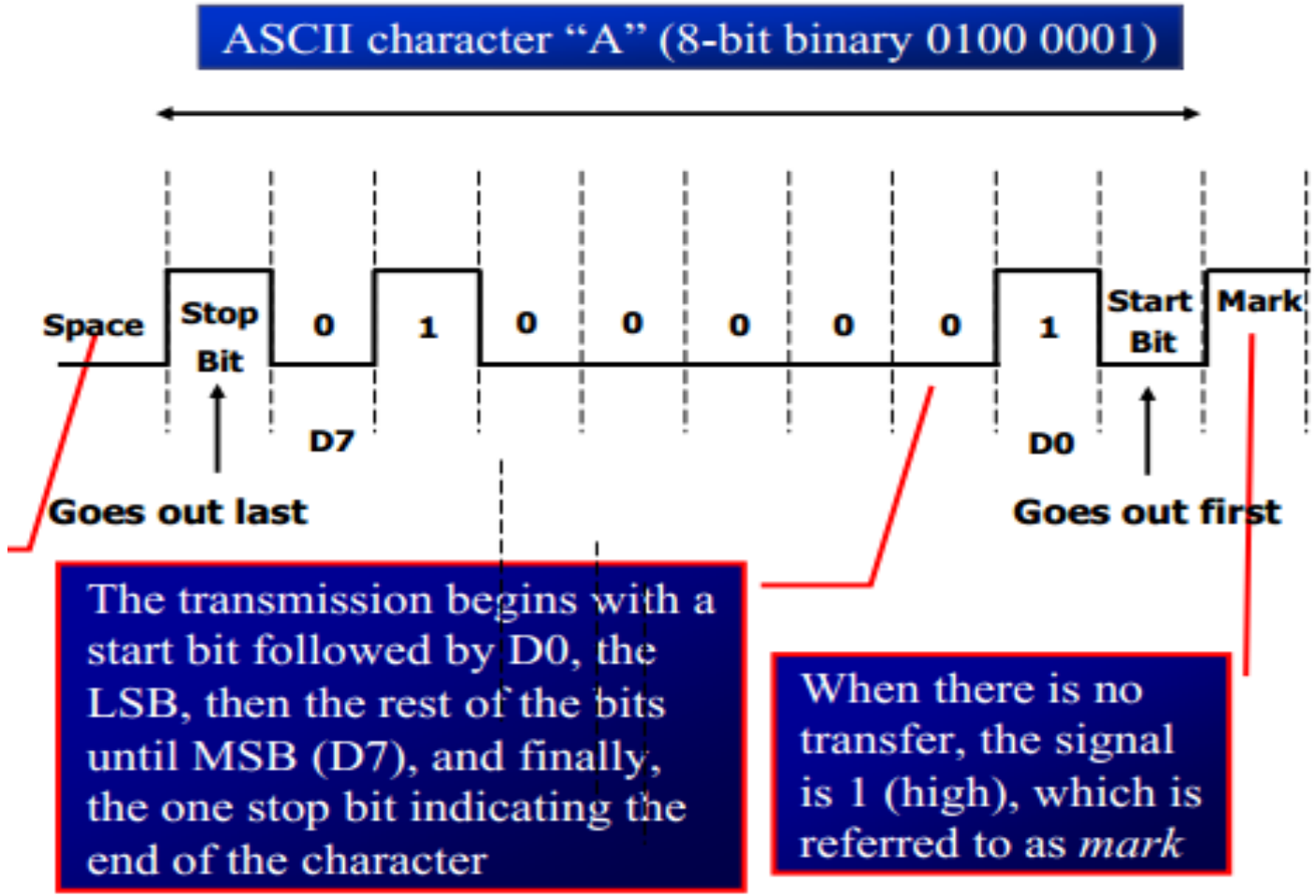
SERIAL COMMUNICATION

- In data transmission, serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus.
- To reduce the number of pins in a package, many ICs use a serial bus to transfer data when speed is not important.
- Some examples of such low-cost serial buses include UART, USART, SPI, I^2C etc.,
- 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 chips made by many manufacturers for serial communications
  - UART (universal asynchronous Receiver-transmitter)
  - USART (universal synchronous-asynchronous Receiver-transmitter)

SERIAL COMMUNICATION



Wiring Connection between two UART



Serial communication message format

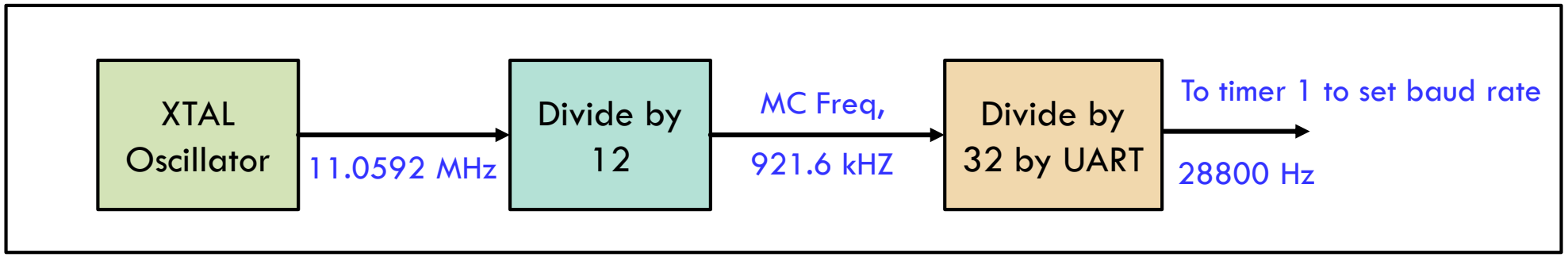
SERIAL COMMUNICATION

- In UART, the start bit is always one bit (HIGH), the stop bit can be one or two bits (LOW).
- The rate of data transfer in serial data communication is stated in bps(bits per second).
- Another widely used terminology for bps is baud rate and is defined as the number of signal changes per second. The baud rate and bps are the same, used inter changeably.
- 8051 has two pins (11 and 10) that are used specifically for transferring (Tx) and receiving(Rx) data serially as a part of the port 3 group (P3.1 and P3.0).
- To allow data transfer between two UART devices, we must make sure that the baud rate of both devices matches.
- Different baud rate levels are, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 etc.,

SERIAL COMMUNICATION

BAUD RATE GENERATION IN 8051

- Dividing 1/12 of the crystal frequency by 32 is the default frequency (28800 Hz) for timers to generate the baud rate.



- With XTAL=11.0592 MHz, find the TH1 value needed to set different baud rate.
  - 28800/3 = 9600 Where -3=FD (hex) is loaded into TH1
  - 28800/6 = 4800 Where -6=FA (hex) is loaded into TH1
  - 28800/12 = 2400 Where -12=F4 (hex) is loaded into TH1
  - 28800/24 = 1200 Where -24=E8 (hex) is loaded into TH1

SERIAL COMMUNICATION

- Register required to work with 8051 serial communications are: SBUF and SCON
- SBUF is an 8-bit register used to hold a data during transmit and receive operation

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

- SCON is an 8-bit the special function register (bit-addressable).
- This register contain not only the mode selection bits but also the 9th data bit for transmit and receive (TB8 and RB8) and the serial port interrupt bits (TI and RI).

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SERIAL COMMUNICATION

BIT	NAME	DESCRIPTION
7	SM0	Serial port mode bit 0
6	SM1	Serial port mode bit 1
5	SM2	Multi processor communication enable bit
4	REN	Receiver Enable. This bit is set in order to receive characters.
3	TB8	Transmit bit 8. The 9^{th} bit to transmit in mode 2 and 3.
2	RB8	Recieve bit 8. The 9^{th} bit to receive in mode 2 and 3.
1	TI	Transmit Interrupt Flag. Set when a byte has been completely transmitted.
0	RI	Receive Interrupt Flag. Set when a byte has been completely Received.

SM0	SM1	MODE	DESCRIPTION	BAUD RATE
0	0	0	Shift register: Serial data are transmitted and received through the RXD pin, while the TXD pin output clocks	$F_{osc.}/12$
0	1	1	8-Bit UART: 8-bit DATA, 1-bit for START, 1-bit for STOP	Variable
1	0	2	9-Bit UART: 9-bit DATA, 1-bit for START, 1-bit for STOP	$F_{osc.}/64$ or $F_{osc.}/32$
1	1	3	9-Bit UART: 9-bit DATA, 1-bit for START, 1-bit for STOP	Variable

SERIAL COMMUNICATION

STEP TO PROGRAM 8051 TO TRANSFER CHARACTER BYTES SERIALLY

1. Load TMOD with 20H, indicating the use of timer 1 in mode 2 (8-bit auto-reload) to set baud rate
2. The TH1 is loaded with one of the values to set baud rate for serial data transfer
3. Load SCON with 50H, indicating mode 1, where an 8-bit data is framed with start and stop bits
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 using JNB TI, xx to check the character is transferred completely or not
7. TI is cleared by CLR TI instruction
8. To transfer the next byte, go to step 5

SERIAL COMMUNICATION

EXAMPLE-1

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

```
        ORG 0000H
        MOV TMOD,#20H      ;timer1,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
```

SERIAL COMMUNICATION

EXAMPLE-2

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

```
        MOV TMOD,#20H       ;timer1,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
```

SERIAL COMMUNICATION

EXAMPLE-3

Assume a switch is connected to pin P1.7. Write an 8051 program to monitor its status and send two messages to serial port continuously as follows. Assume XTAL = 11.0592 MHz, 9600 baud, 8-bit data, and 1 stop bit.

* SW=0 send "NO"
* SW=1 send "YES"

```
        ORG 0000H           ;starting position
MAIN:   MOV TMOD, #20H      ;timer1,mode 2(auto reload)
        MOV TH1,#-3         ; 9600 baud rate
        MOV SCON, #50H      ;8-bit, 1 stop, REN enabled
        SETB TR1            ; start timer
        SETB P1.7           ; make SW an input
S1:     JB P1.7, NEXT       ; check SW status
        MOV DPTR, #MESS1    ; if SW=0 display "NO"
FN:     CLR A
        MOVC A, @A+DPTR     ; read the value; check for end of line
        JZ S1               ; check for end of line
        ACALL SENDCOM       ; send value to serial port
```

SERIAL COMMUNICATION

```
                INC DPTR            ; move to next value
                SJMP FN             ; repeat
NEXT:           MOV DPTR, #MESS2    ; if SW=1 display "YES“
LN:             CLR A
                MOVC A, @A+DPTR     ; read the value; check for end of line ;
                JZ S1               ; check for end of line
                ACALL SENDCOM       ; send value to serial port
                INC DPTR            ; move to next value
                SJMP LN             ; repeat
SENDCOM:        MOV SBUF, A         ; place the value in SBUF
HERE:           JNB TI, HERE        ; wait until transmit complete
                CLR TI              ; clear TI
                RET                 ; return
MESS1:          DB ‘NO’,0
MESS2:          DB ‘YES’, 0
                END
```

SERIAL COMMUNICATION

STEP TO PROGRAM 8051 TO RECEIVE CHARACTER BYTES SERIALLY

1. Load TMOD with 20H, indicating the use of timer 1 in mode 2 (8-bit auto-reload) to set baud rate
2. The TH1 is loaded with one of the values to set baud rate for serial data transfer
3. Load SCON with 50H, indicating mode 1, where an 8-bit data is framed with start and stop bits
4. TR1 is set to 1 to start timer 1
5. The RI flag bit is monitored using JNB RI, xx to check the character is received completely or not
6. If RI is raised, SBUF register has a byte and transfer it to accumulator
7. RI is cleared by CLR RI instruction
8. To transfer the next byte, go to step 5

SERIAL COMMUNICATION

EXAMPLE-4

Write a program for the 8051 to receive bytes of data serially, and put them in P1, set the baud rate at 4800, 8-bit data, and 1 stop bit.

```
        ORG 0000H
        MOV P1,#00H
        MOV TMOD,#20H       ;timer1,mode 2(auto reload)
        MOV TH1,#-6          ;4800 baud rate
        MOV SCON,#50H       ;8-bit, 1 stop, REN enabled
        SETB TR1            ;start timer 1
HERE:   JNB RI,HERE         ;wait for char to come in
        MOV A,SBUF          ;saving incoming byte in A
        MOV P1,A            ;send to port 1
        CLR RI              ;get ready to receive next byte
        SJMP HERE           ;keep getting data
```

SERIAL COMMUNICATION

EXAMPLE-5

Assume that the 8051 serial port is connected to the COM port of IBM PC, and on the PC, we are using the terminal.exe program to send and receive data serially. P1 and P2 of the 8051 are connected to LEDs and switches, respectively. Write an 8051 program to

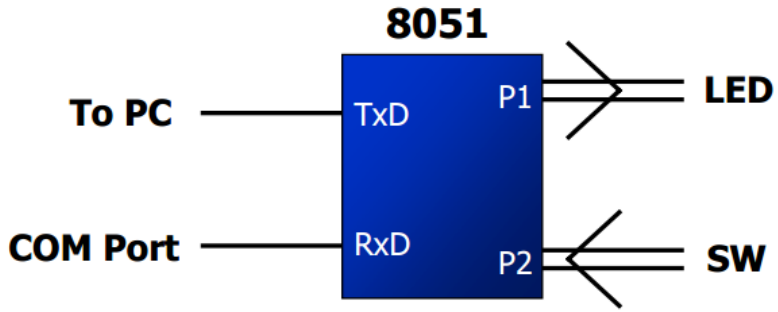
(a) send to PC the message "We Are Ready",

(b) receive any data send by PC and put it on LEDs connected to P1, and

(c) get data on switches connected to P2 and send it to PC serially.

The program should perform part (a) once, but parts (b) and (c) continuously, use 4800 baud rate.

```
ORG 0000H
MOV P1,#00H
MOV P2,#0FFH        ;make P2 an input port
MOV TMOD,#20H       ;timer 1, mode 2
MOV TH1,#0FAH       ;4800 baud rate
MOV SCON,#50H       ;8-bit, 1 stop, REN enabled
SETB TR1            ;start timer 1
```



SERIAL COMMUNICATION

```
        MOVC DPTR,#MYDATA   ;load pointer for message
H_1:    CLR A
        MOV A,@A+DPTR       ;get the character
        JZ B_1              ;if last character get out
        ACALL SEND          ;otherwise call transfer
        INC DPTR            ;next one
        SJMP H_1            ;stay in loop
B_1:    MOV A,P2            ;read data on P2
        ACALL SEND          ;transfer it serially
        ACALL RECV          ;get the serial data
        MOV P1,A            ;display it on LEDs
        SJMP B_1            ;stay in loop indefinitely
```

SERIAL COMMUNICATION

```
;----serial data transfer. ACC has the data------
SEND:       MOV SBUF,A      ;load the data
H_2:        JNB TI,H_2      ;stay here until last bit gone
            CLR TI          ;get ready for next char
            RET             ;return to caller
;----Receive data serially in ACC----------------
RECV:       JNB RI,RECV     ;wait here for char
            MOV A,SBUF      ;save it in ACC
            CLR RI           ;get ready for next char
            RET             ;return to caller
;-----The message--------------
MYDATA:     DB "We Are Ready",0
            END
```

Exercise: Write a program to send the message "The Earth is but One Country" to serial port. Assume a SW is connected to pin P1.2. Monitor its status and set the baud rate as follows:

- SW = 0, 4800 baud rate
- SW = 1, 9600 baud rate

Assume XTAL = 11.0592 MHz, 8-bit data, and 1 stop bit

INTERRUPTS

INTERRUPTS

- A single microcontroller can serve several devices by two ways: (i) Interrupt (ii). Polling
- Polling can monitor the status of several devices and serve each of them as certain conditions are met
  - The polling method is not efficient, since it wastes much of the microcontroller's time by polling devices that do not need service, ex. JNB TF, target
- 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
- The advantage of interrupts is that the microcontroller can serve many devices (not all at the same time).

INTERRUPTS

- Upon activation of an interrupt, the microcontroller goes through the following steps:

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack
2. It also saves the current status of all the interrupts internally (i. e: not on the stack)
3. It jumps to a fixed memory location called interrupt vector table, that holds the address of the ISR
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it
5. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt)
6. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted
7. First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC
8. Then it starts to execute from that address

INTERRUPTS

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	

```
        ORG   0       ;wake-up ROM reset location
        LJMP  MAIN    ;by-pass int. vector table
;---- the wake-up program
        ORG   30H
MAIN:
        ....
        END
```

Only three bytes of ROM space assigned to the reset pin. We put the LJMP as the first instruction and redirect the processor away from the interrupt vector table.

INTERRUPTS

INTERRUPT ENABLE (IE) REGISTER

BIT	NAME	DESCRIPTION
7	EA	Enable All must be set to 1 in order activate each interrupt given in the register
6	--	Reserved for future use
5	ET2	Enable/Disable Timer 2 overflow interrupt (for 8952)
4	ES	Enable/Disable Serial port interrupt
3	ET1	Enable/Disable Timer 1 overflow interrupt
2	EX1	Enable/Disable eXternal interrupt 1
1	ET0	Enable/Disable Timer 1 overflow interrupt
0	EX0	Enable/Disable eXternal interrupt 0

INTERRUPTS

- The timer flag (TF) is raised when the timer rolls over
  - In polling TF, we have to wait until the TF is raised
  - The problem with this method is that the microcontroller is tied down while waiting for TF to be raised, it can't do anything else
  - Using interrupts solves this problem and, avoids tying down the controller
  - If the timer interrupt in the IE register

INTERRUPTS

EXAMPLE-1

Write a program that continuously using interrupts to get 8-bit data from P0 and sends it to P1 while simultaneously creating a square wave of 200 µs period on pin P2.1. Use timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

Delay calculation: Use timer 0 in mode 2 (auto reload). TH0 = 100/1.085 us = 92 =A4H

```
;--upon wake-up go to main, avoid using memory allocated to Interrupt Vector Table
        ORG 0000H
        LJMP MAIN               ;by-pass interrupt vector table

;--ISR for timer 0 to generate square wave
        ORG 000BH               ;Timer 0 interrupt vector table
        CPL P2.1                ;toggle P2.1 pin
        RETI                    ;return from ISR
```

INTERRUPTS

```
;--The main program for initialization
                ORG 0030H           ;after vector table space
MAIN:           MOV P0,#0FFH        ;make P0 an input port
                MOV P1,#00H         ;make P0 an output port
                CLR P2.1            ;make P2.1 as output pin
                MOV TMOD,#02H       ;Timer 0, mode 2
                MOV TH0,#0A4H       ;TH0=A4H for -92
                MOV IE,#82H         ;IE=10000010 (bin) enable Timer 0
                SETB TR0            ;Start Timer 0
BACK:           MOV A,P0            ;get data from P0
                MOV P1,A            ;issue it to P1
                SJMP BACK           ;keep doing it loop unless interrupted by TF0
                END
```

INTERRUPTS

- The 8051 has two external hardware interrupts, Pin 12 (P3.2) and pin 13 (P3.3) of the 8051, designated as INT0 and INT1
- The interrupt vector table locations 0003H and 0013H are set aside for INT0 and INT1
- There are two activation levels for the external hardware interrupts

1. In Level trigged Mode, INT0 and INT1 pins are normally high, if a low-level signal is applied to them, it triggers the interrupt

2. Edge trigged Mode:

   - To make INT0 and INT1 edge-triggered (falling edge) interrupts, we must program the bits of the TCON register bit TCON.2 (IT1) and TCON.0 (IT0).
   - The falling edge of pins INT0 and INT1 are latched by the 8051 and are held by the TCON.1 and TCON.3 bits of TCON register.

INTERRUPTS

TCON REGISTER

- TFx: Timerx Overflow Flag
  - TFx =1 means Timerx overflow occurred (i.e. Timerx goes to its max and roll over back to zero).
  - TFx =1 means Timerx overflow not occurred.

- TRx: Timerx Run Control Bit
  - TRx =1 means Timerx start.
  - TRx =0 means Timerx stop.

- IEx: External Interruptx Edge Flag
  - IEx=1 means External interruptx occurred.
  - IEx=0 means External interruptx Processed.

- ITx: External Interruptx Trigger Type Select Bit
  - ITx=1 means Interrupt occurs on falling edge at INTx pin.
  - ITx=0 means Interrupt occur on a low level at the INTx pin.

*Where x represent 0 for Timer0 and 1 for Timer 1

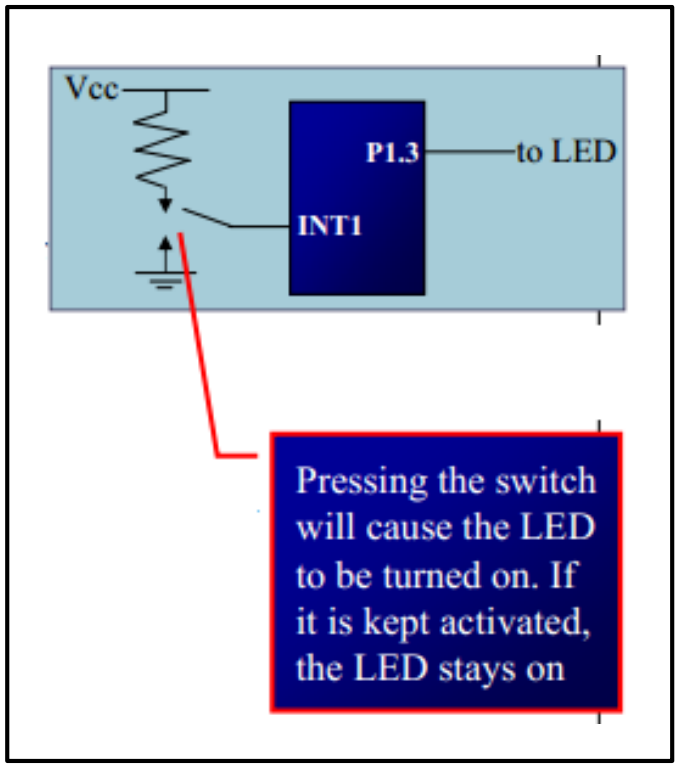
INTERRUPTS

EXAMPLE-2

Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to P1.3 and is normally off. As long as the switch is pressed low, the LED should stay on. Simultaneously perform a toggle operation in P1.5 with the delay of 500ms.

```
        ORG 0000H
        LJMP MAIN

//ISR for INT1
        ORG 0013H
        SETB P1.3
        RETI
```



INTERRUPTS

```
        ORG 30H
MAIN:   SETB P3.3
        CLR P1.3
        CLR P1.5
        MOV IE,#10000100B
HERE:   CLR P1.3
        SETB P1.5
        ACALL DELAY
        CLR P1.5
        ACALL DELAY
        SJMP HERE
```

```
//Delay of 500ms
DELAY:  MOV R2,#04H      ;LOAD R2 WITH 07 HEX
HERE3:  MOV R1,#0FFH     ;LOAD R1 WITH 0FF HEX
HERE2:  MOV R0,#0FFH     ;LOAD R2 WITH 0FF HEX
HERE1:  DJNZ R0,HERE1    ;DECREMENT R0
        DJNZ R1,HERE2    ;DECREMENT R1
        DJNZ R2,HERE3    ;DECREMENT R2
        RET              ;RETURN
        END
```

INTERRUPTS

EXAMPLE-3

Assume that 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, which is connected to an LED (or buzzer). In other words, the LED is turned on and off at the same rate as the *pulses* are applied to the INT1 pin.

```
        ORG 0000H
        LJMP MAIN
;--ISR for hardware interrupt INT1 to turn on LED
        ORG 0013H       ;INT1 ISR
        SETB P1.3       ;turn on LED
        RETI            ;return from ISR
```

```
;------MAIN program for initialization
        ORG 30H
        SETB P3.3       ; make P3.3 as input
        CLR P1.3        ; make p1.3 as output
MAIN:   SETB TCON.2     ;make INT1 edge-triggered.
        MOV IE,#10000100B ;enable External INT 1
HERE:   CLR P1.3            ;turn off the buzzer
        SJMP HERE       ;stay here until get interrupted
        END
```

Exercise: Write a program in which the 8051 reads data from P1 and writes it to P2 continuously while giving a copy of it to the serial COM port to be transferred serially. Assume that XTAL=11.0592. Set the baud rate at 9600.

INTERRUPTS

EXAMPLE-4

Write a program using interrupts to do the following:

(a) Receive data serially and sent it to P0,

(b) Have P1 port read and transmitted serially, and a copy given to P2,

(c) Make timer 0 generate a square wave of 5kHz frequency on P0.1.

Assume that XTAL-11,0592. Set the baud rate at 4800.

```
        ORG 0
        LJMP MAIN

        ORG 000BH           ;ISR for timer 0
        CPL P0.1            ;toggle P0.1
        RETI                ;return from ISR

        ORG 23H
        LJMP SERIAL         ;jump to serial interrupt ISR
```

INTERRUPTS

```
        ORG 30H
MAIN:   MOV P0,#00H         ;make P0 an output port
        MOV P1,#0FFH        ;make P1 an input port
        MOV P2,#00H         ;make P2 an output port
        MOV TMOD,#22H       ;timer 1,mode 2(auto reload)
        MOV TH1,#0F6H       ;4800 baud rate
        MOV SCON,#50H       ;8-bit, 1 stop, REN enabled
        MOV TH0,#-92        ;for 5kHZ wave
        MOV IE,10010010B    ;enable serial int.
        SETB TR1            ;start timer 1
        SETB TR0            ;start timer 0
BACK:   MOV A,P1            ;read data from port 1
        MOV SBUF,A          ;give a copy to SBUF
        MOV P2,A            ;send it to P2
        SJMP BACK           ;stay in loop indefinitely
```

INTERRUPTS

```
        ORG 100H
SERIAL: JB TI,TRANS     ;jump if TI is high
        MOV A,SBUF      ;otherwise due to receive
        MOV P0,A        ;send serial data to P0
        CLR RI          ;clear RI since CPU doesn't
        RETI            ;return from ISR
TRANS:  CLR TI          ;clear TI since CPU doesn't
        RETI            ;return from ISR
        END
```

Exercise: Write a 8051 assembly program using interrupts to do the following:

(a) Generate a 10 KHz frequency on P2.1

(b) Use timer 1 as an event counter to count and display it on P0. The pulse is connected to EX1.

Assume that XTAL = 11.0592 MHz. Set the baud rate at 9600.

THANK YOU

by
prakash v