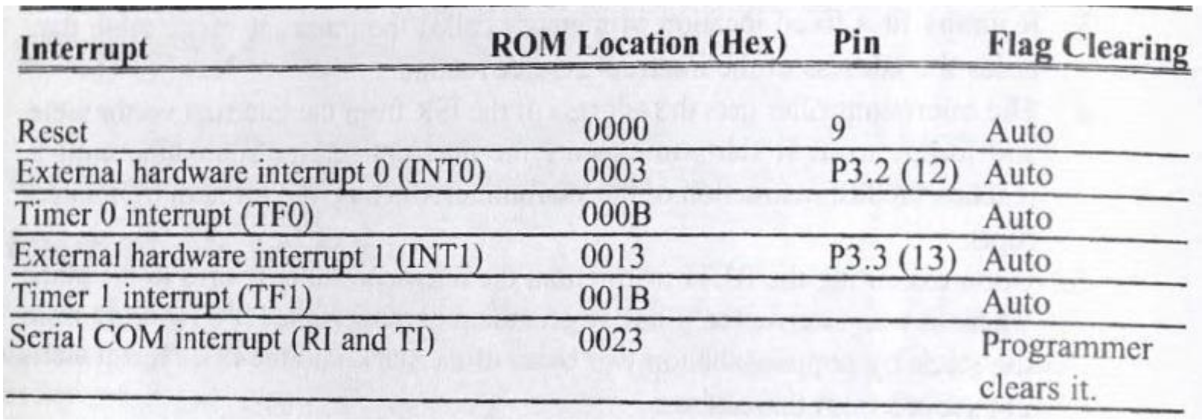
INTERRUPT PROGRAMMING IN 8051

The Microcontroller can serve several devices. The Interrupt is the method to indicate the microcontroller by sending an interrupt signal. After receiving an interrupt, the microcontroller interrupts whatever it is doing and serves the device. The program associated with the interrupt is called the interrupt service routine (ISR). When an interrupt is invoked, the microcontroller runs the interrupt service routine. For every interrupt, there is a fixed location set aside to hold the addresses of ISRs.

The following events will cause an interrupt:

1. Timer 0 Overflow.
2. Timer 1 Overflow.
3. Reception/Transmission of Serial Character.
4. External Event 0.
5. External Event 1.

To distinguish between various interrupts and executing different code depending on what interrupt was triggered, 8051may be jumping to a fixed address when a given interrupt occurs as shown in Table 5.3.1.



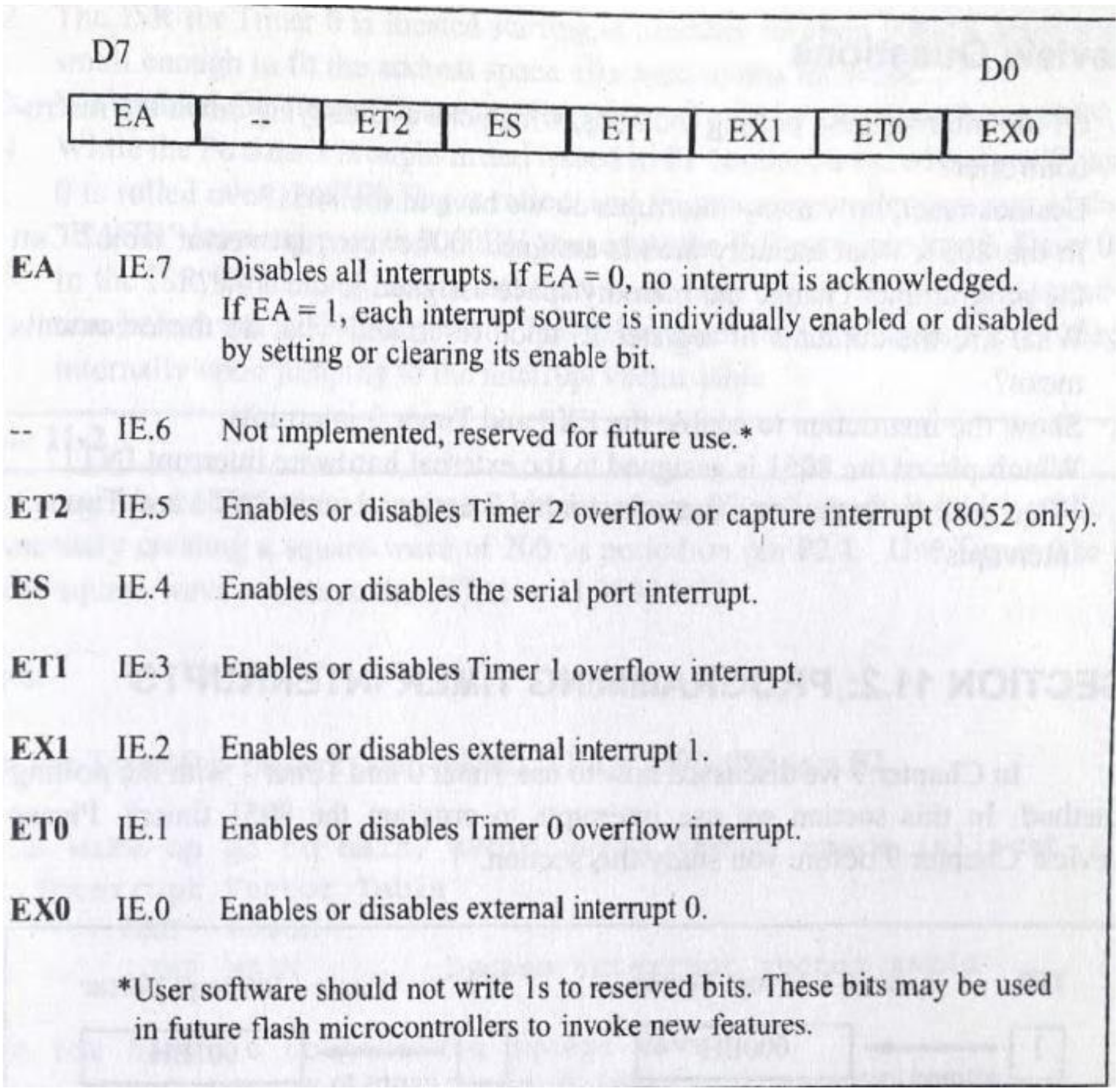
Interrupt	ROM Location (Hex)	Pin	Flag Clearing
Reset	0000	9	Auto
External hardware interrupt 0 (INT0)	0003	P3.2 (12)	Auto
Timer 0 interrupt (TF0)	000B		Auto
External hardware interrupt 1 (INT1)	0013	P3.3 (13)	Auto
Timer 1 interrupt (TF1)	001B		Auto
Serial COM interrupt (RI and TI)	0023		Programmer clears it.

Table 5.3.1 Interrupt Vector Table for 8051

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.320]

ENABLING AND DISABLING AN INTERRUPT

Upon reset all interrupts are disable, meaning that known will be responded to by the microcontroller if they are activated. The Interrupt must be enabled by software in order for microcontroller to respond to them there is a register called IE that is responsible for enabling and disabling the interrupts as shown in Figure 5.3.1



D7 D0

EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
--	IE.6	Not implemented, reserved for future use.*
ET2	IE.5	Enables or disables Timer 2 overflow or capture interrupt (8052 only).
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.

*User software should not write 1s to reserved bits. These bits may be used in future flash microcontrollers to invoke new features.

Figure 5.3.1 Interrupt Enable(IE) Register

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.321]

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

The 8051 has two external hardware interrupts PIN 12 (P3.2) and Pin 13 (P3.3), designated as INT0 and INT1. Upon activation of these pins, the 8051 finishes the execution of current instruction whatever it is executing and jumps to the vector table to perform the interrupt service routine.

TYPES OF INTERRUPT

1) Level-TriggeredInterrupt
2) Edge -Triggered Interrupt

LEVEL-TRIGGERED INTERRUPT

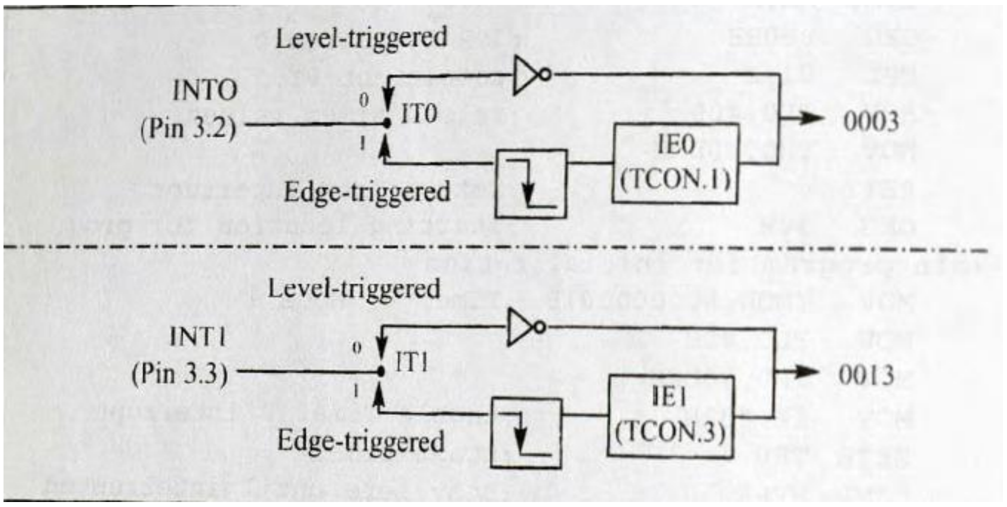
In this mode, INT0 and INT1 are normally high and if the low level signal is applied to them, it triggers the Interrupt. Then the microcontroller stops and jumps to the interrupt vector table to service that interrupt. The low-level signal at the INT pin must be removed before the execution of the last instruction of the ISR, RETI. Otherwise, another interrupt will be generated. This is called a level-triggered or level-activated interruptandis the default mode upon reset

Figure 5.3.2 Activation of INT0 and INT1

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.326]

EDGE -TRIGGERED INTERRUPT

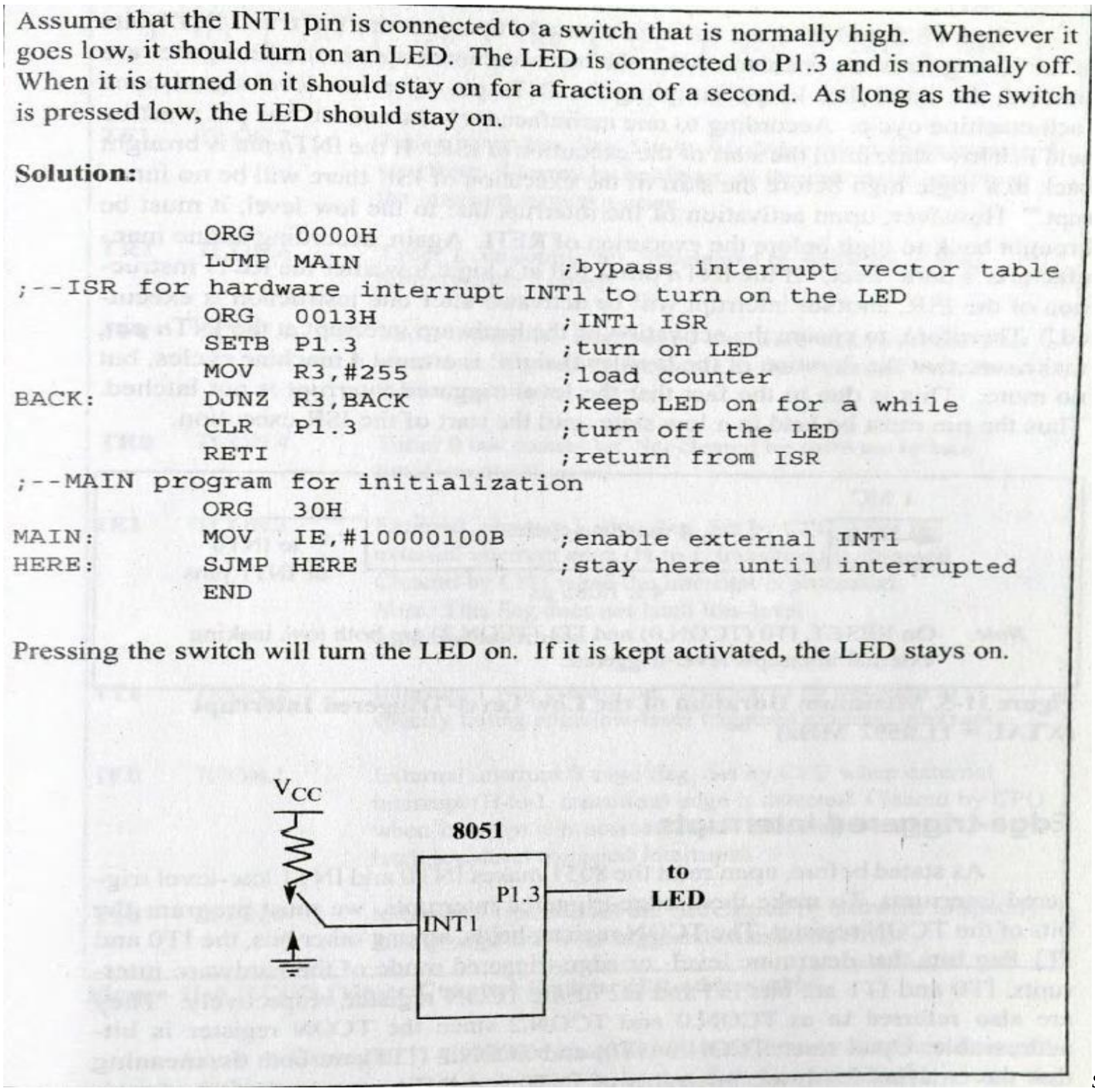
Upon reset 8051 makes INT0 and INT1 low l Level-Triggered Interrupt. To make them Edge -Triggered Interrupt, we must program the bits of the TCON Register. The



TCON register holds among other bits and IT0 and IT1 flags bit the determine level- or edge triggered mode. IT0 and IT1 are bits D0 (TCON.0) and D2(TCON.2) of the TCON Register respectively.

Figure 5.3.3 Example for Level triggered Interrupt

[Source: "The 8051Microcontroller and Embedded Systems: U

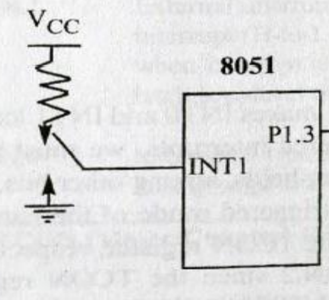


Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to P1.3 and is normally off. When it is turned on it should stay on for a fraction of a second. As long as the switch is pressed low, the LED should stay on.

Solution:

```
        ORG   0000H
        LJMP  MAIN             ;bypass interrupt vector table
;--ISR for hardware interrupt INT1 to turn on the LED
        ORG   0013H            ;INT1 ISR
        SETB  P1.3             ;turn on LED
        MOV   R3,#255          ;load counter
BACK:   DJNZ  R3,BACK          ;keep LED on for a while
        CLR   P1.3             ;turn off the LED
        RETI                   ;return from ISR
;--MAIN program for initialization
        ORG   30H
MAIN:   MOV   IE,#10000100B    ;enable external INT1
HERE:   SJMP  HERE             ;stay here until interrupted
        END
```

Pressing the switch will turn the LED on. If it is kept activated, the LED stays on.



sing Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.327]

SERIAL COMMUNICATION INTERRUPT

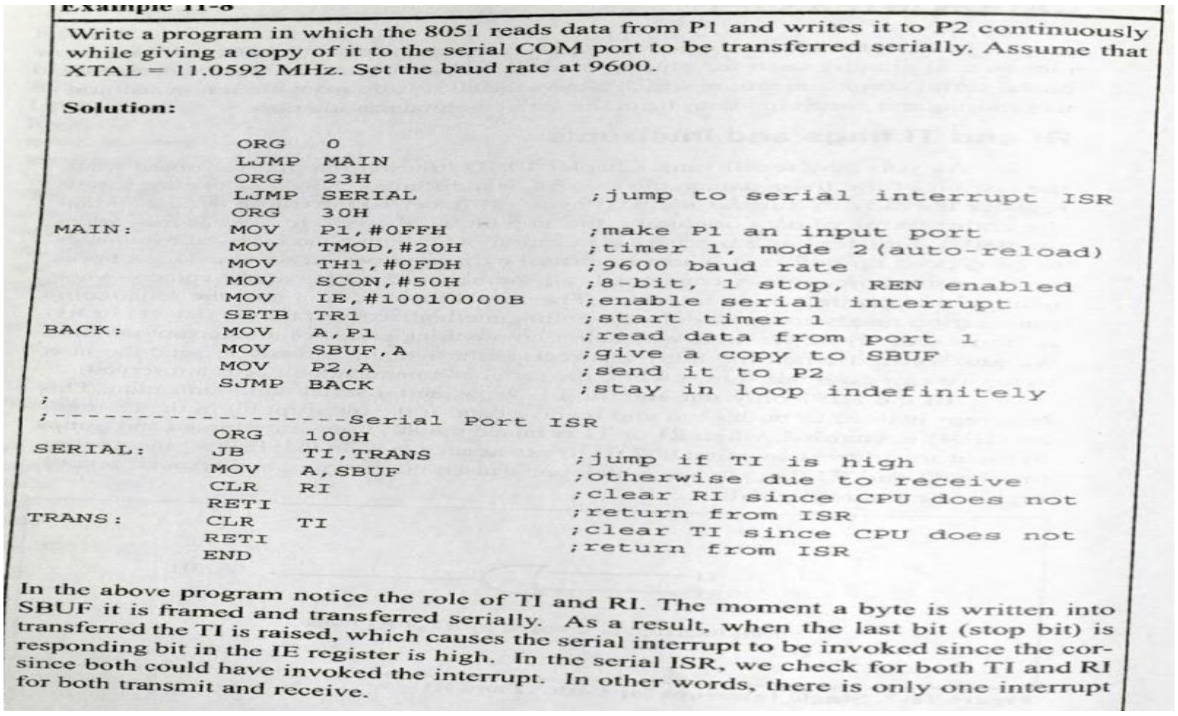
TI (transfer interrupt) is raised when the stop bit is transferred indicating that the SBUF register is ready to transfer the next byte

RI (received interrupt) is raised when the stop bit is received indicating that the received byte needs to be picked up before it is lost (overrun) by new incoming serial data

In the 8051 there is only one interrupt set aside for serial communication ,used for both sending and receiving data.

If the interrupt bit in the IE register (IE.4) is enabled, when RI or TI is raised the 8051 gets interrupted and jumps to memory location 0023H to execute the ISR

In that ISR we must examine the TI and RI flags to see which one caused the interrupt



Example 11-8

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 MHz. Set the baud rate at 9600.

Solution:

```
          ORG   0
          LJMP  MAIN
          ORG   23H
          LJMP  SERIAL             ;jump to serial interrupt ISR
          ORG   30H
MAIN:     MOV   P1,#0FFH           ;make P1 an input port
          MOV   TMOD,#20H          ;timer 1, mode 2(auto-reload)
          MOV   TH1,#0FDH          ;9600 baud rate
          MOV   SCON,#50H          ;8-bit, 1 stop, REN enabled
          MOV   IE,#10010000B      ;enable serial interrupt
          SETB  TR1                ;start timer 1
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
;
;-------------------Serial Port ISR
          ORG   100H
SERIAL:   JB    TI,TRANS           ;jump if TI is high
          MOV   A,SBUF             ;otherwise due to receive
          CLR   RI                 ;clear RI since CPU does not
          RETI                     ;return from ISR
TRANS:    CLR   TI                 ;clear TI since CPU does not
          RETI                     ;return from ISR
          END
```

In the above program notice the role of TI and RI. The moment a byte is written into SBUF it is framed and transferred serially. As a result, when the last bit (stop bit) is transferred the TI is raised, which causes the serial interrupt to be invoked since the corresponding bit in the IE register is high. In the serial ISR, we check for both TI and RI since both could have invoked the interrupt. In other words, there is only one interrupt for both transmit and receive.

and respond accordingly.

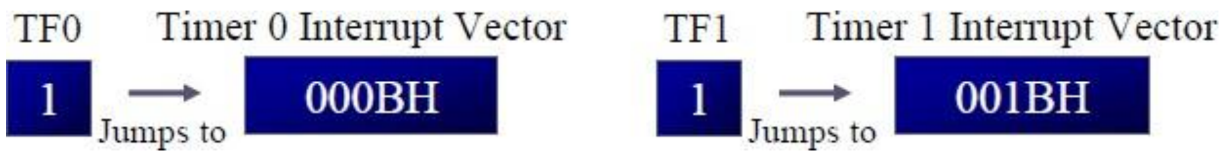
Figure 5.3.4 Example for Serial Communication Interrupt

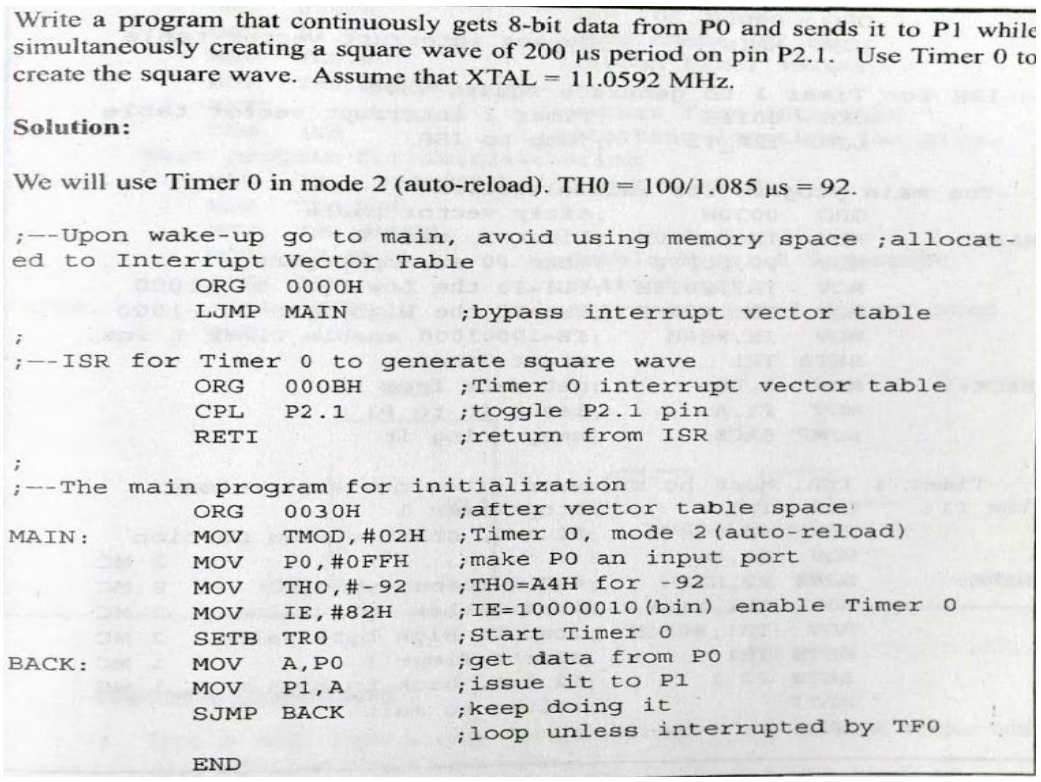
[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.334]

TIMER 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 microcontroller is tied down while waiting for TF to be raised, and cannot do anything else. If the timer interrupt in the IE register is enabled, whenever the timer rolls over, TF is raised. This avoids tying down the controller.

The microcontroller is interrupted in whatever it is doing, and jumps to the interrupt vector table to service the ISR.In this way, the microcontroller can do other task until it is notified that the timer has rolled over





Write a program that continuously gets 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.

Solution:

We will use Timer 0 in mode 2 (auto-reload). TH0 = 100/1.085 μs = 92.

```
;--Upon wake-up go to main, avoid using memory space ;allocat-
ed to Interrupt Vector Table
          ORG   0000H
          LJMP  MAIN         ;bypass 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
;
;--The main program for initialization
          ORG   0030H        ;after vector table space
MAIN:     MOV   TMOD,#02H    ;Timer 0, mode 2(auto-reload)
          MOV   P0,#0FFH     ;make P0 an input port
          MOV   TH0,#-92     ;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
```

Figure 5.3.5 Example for Timer Interrupt

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.323]