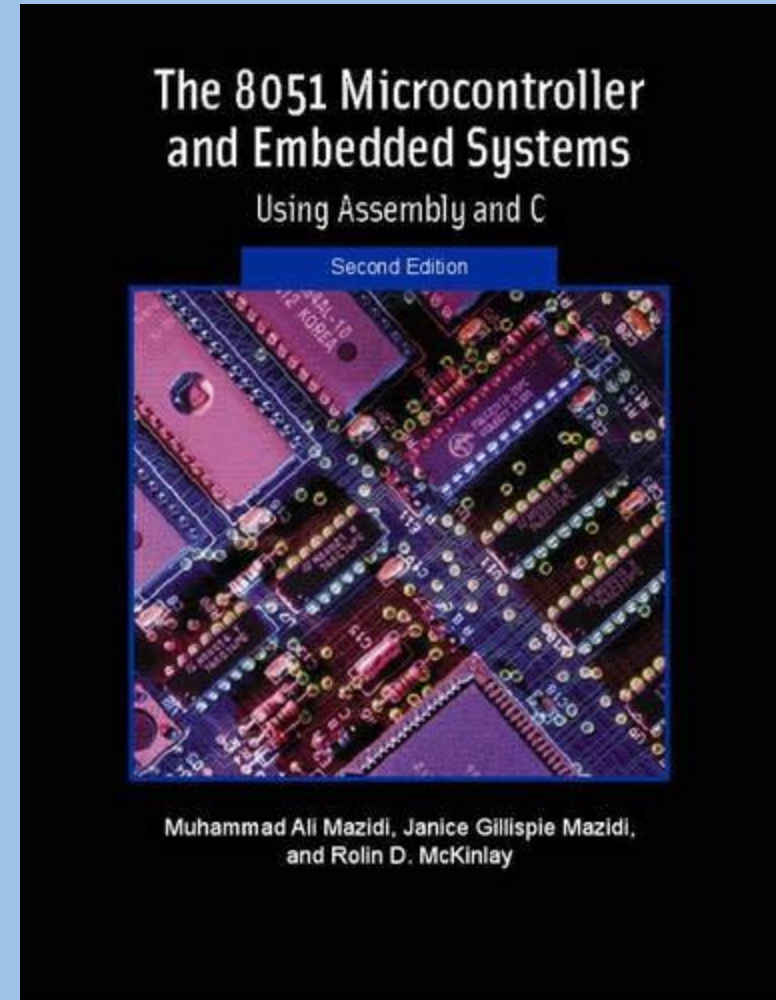


THE 8051 MICROCONTROLLER & Embedded Systems

Muhammad Ali Mazidi, Janice Mazidi
& Rolin McKinlay



INTERRUPTS PROGRAMMING IN ASSEMBLY AND C

Chapter 11

Objectives

Upon completion of this chapter, you will be able to:

- >> Contrast and compare interrupts versus polling**
- >> Explain the purpose of the ISR (interrupt service routine)**
- >> List the 6 interrupts of the 8051**
- >> Explain the purpose of the interrupt vector table**
- >> Enable or disable 8051/52 interrupts**
- >> Program the 8051/52 timers using interrupts**
- >> Describe the external hardware interrupts of the 8051/52**
- >> Contrast edge-triggered with level-triggered interrupts**
- >> Program the 8051 for interrupt-based serial communication**
- >> Define the interrupt priority of the 8051**
- >> Program 8051/52 interrupts in C**

Objectives

- **The concept of the interrupt and interrupt programming**
- **Section 11.1, the basics of 8051 interrupts,**
- **Section 11.2, the interrupts belonging to Timers 0 and 1,**
- **Section 11.3, external hardware interrupts,**
- **Section 11.4, the interrupt related to serial communication,**
- **Section 11.5, interrupt priority in the 8051/52,**
- **Section 11.6, C programming of 8051 interrupts**

Table 11-1: Interrupt Vector Table for the 8051

<u>Interrupt</u>	<u>ROM Location (Hex)</u>	<u>Pin</u>	<u>Flag Clearing</u>
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.

Figure 11-1. Redirecting the 8051 from the Interrupt Vector Table at Power-up

```
                ORG 0      ;wake-up ROM reset location
                LJMP MAIN ;bypass interrupt vector
table

;----- the wake-up program
                ORG 30H
MAIN:
                . . . .
                END
```

Figure 11-2. IE (Interrupt Enable) Register

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.

Example 11-1

Show the instructions to (a) enable the serial interrupt, Timer 0 interrupt, and external hardware interrupt 1 (EX1), 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

Since IE is a bit-addressable register, we can use the following instructions to access individual bits of the register.

(b) `CLR IE.1` ;mask(disable) Timer 0 interrupt only

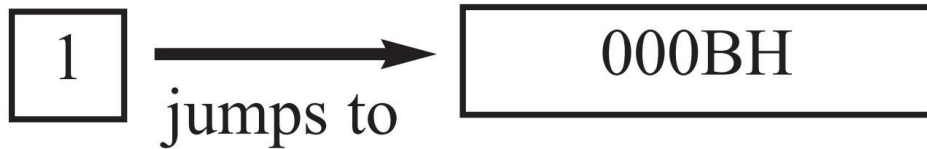
(c) `CLR IE.7` ;disable all interrupts

Another way to perform the “`MOV IE,#10010110B`” instruction is by using single-bit instructions as shown below.

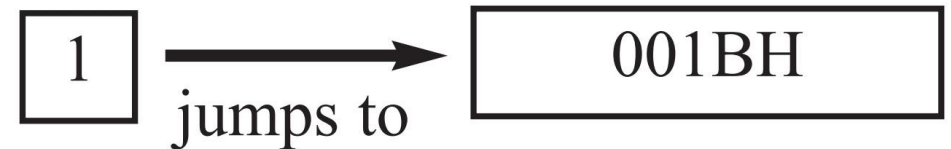
<code>SETB IE.7</code>	;EA=1, Global enable
<code>SETB IE.4</code>	;enable serial interrupt
<code>SETB IE.1</code>	;enable Timer 0 interrupt
<code>SETB IE.2</code>	;enable EX1

Figure 11-3. TF Interrupt

TF0 Timer 0 Interrupt Vector



TF1 Timer 1 Interrupt Vector



Example 11-2

Solution:

We will use Timer 0 in mode 2 (auto-reload). $TH0 = 100/1.085 \text{ ms} = 92$.

;
--Upon wake-up go to main, avoid using memory space ;allocated 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
```

Write a program that continuously gets 8-bit data from P0 and sends it to P1 while simultaneously creating a square wave of 200 ms period on pin P2.1.

Use Timer 0 to create the square wave.

Assume that
XTAL = 11.0592 MHz.

Example 11-3

Rewrite Example 11-2 to create a square wave that has a high portion of 1085 μs and a low portion of 15 μs .

Assume
XTAL = 11.0592 MHz.

Use Timer 1.

Solution:

Since 1085 μs is 1000×1.085 we need to use mode 1 of Timer 1.

--Upon wake-up go to main, avoid using memory space
--allocated to Interrupt Vector Table

```
ORG      0000H
LJMP     MAIN      ;bypass interrupt vector table

;
;--ISR for Timer 1 to generate square wave
ORG      001BH      ;Timer 1 interrupt vector table
LJMP     ISR_T1     ;jump to ISR

;
;--The main program for initialization
ORG      0030H      ;after vector table
MAIN:    MOV        TMOD,#10H      ;Timer 1, mode 1
          MOV        P0,#0FFH      ;make P0 an input port
          MOV        TL1,#018H     ;TL1=18 the Low byte of -1000
          MOV        TH1,#0FCH     ;TH1=FC the High byte of -1000
          MOV        IE,#88H      ;IE=10001000 enable Timer 1 int.
          SETB       TR1          ;start Timer 1
BACK:    MOV        A,P0          ;get data from P0
          MOV        P1,A         ;issue it to P1
          SJMP       BACK         ;keep doing it

;
;--Timer 1 ISR. Must be reloaded since not auto-reload
ISR_T1:  CLR        TR1          ;stop Timer 1
          CLR        P2.1        ;P2.1=0, start of low portion
          MOV        R2,#4        ;
          2 MC
HERE:    DJNZ       R2,HERE      ;4x2 machine cycle(MC)          8 MC
          MOV        TL1,#18H     ;load T1 Low byte value       2 MC
          MOV        TH1,#0FCH    ;load T1 High byte value      2 MC
          SETB       TR1          ;starts Timer 1              1 MC
          SETB       P2.1        ;P2.1=1, back to high         1 MC
          RETI                  ;return to main
END
```

Example 11-4

Write a program to generate a square wave of 50 Hz frequency on pin P1.2. This is similar to Example 9-12 except that it uses an interrupt for Timer 0. Assume that XTAL = 11.0592 MHz.

Solution:

```
ORG      0
LJMP     MAIN
ORG      000BH           ;ISR for Timer 0
CPL      P1.2           ;complement P1.2
MOV      TL0,#00         ;reload timer values
MOV      TH0,#0DCH
RETI                      ;return from interrupt
ORG      30H             ;starting location for prog.
;-----main program for initialization
MAIN:    MOV      TMOD,#00000001B      ;Timer 0, Mode 1
          MOV      TL0,#00
          MOV      TH0,#0DCH
          MOV      IE,#82H             ;enable Timer 0 interrupt
          SETB     TR0                 ;start timer
HERE:    SJMP     HERE                 ;stay here until interrupted
END
```

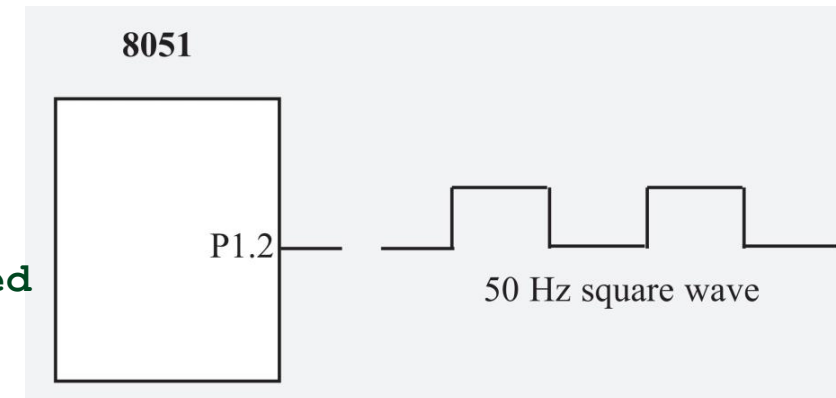
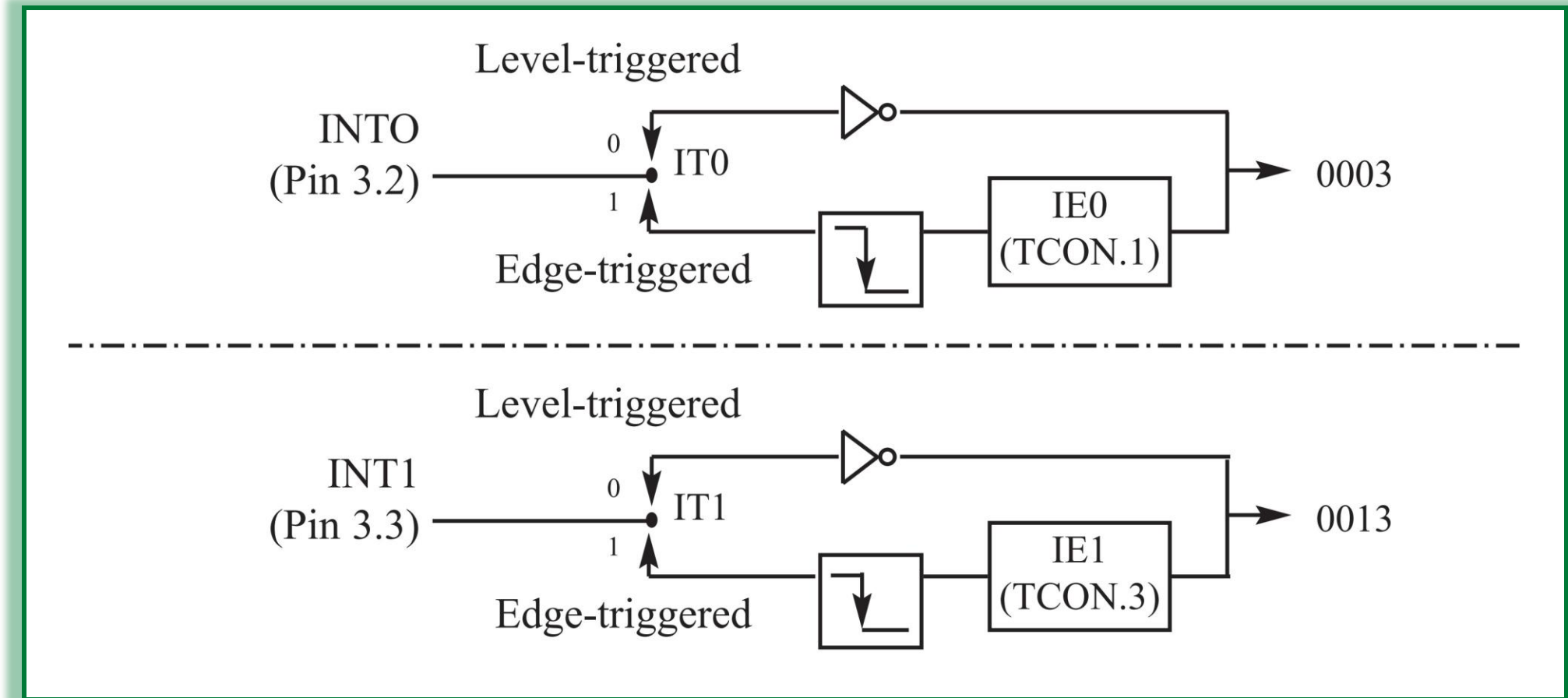


Figure 11-4. Activation of INT0 and INT1



Example 11-5

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.

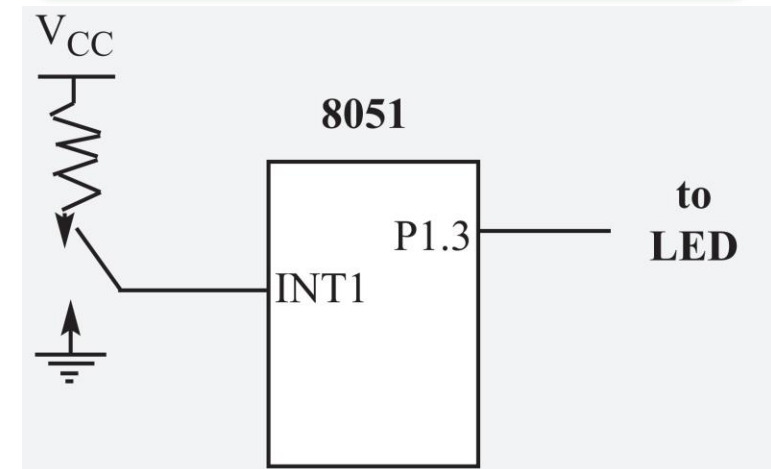
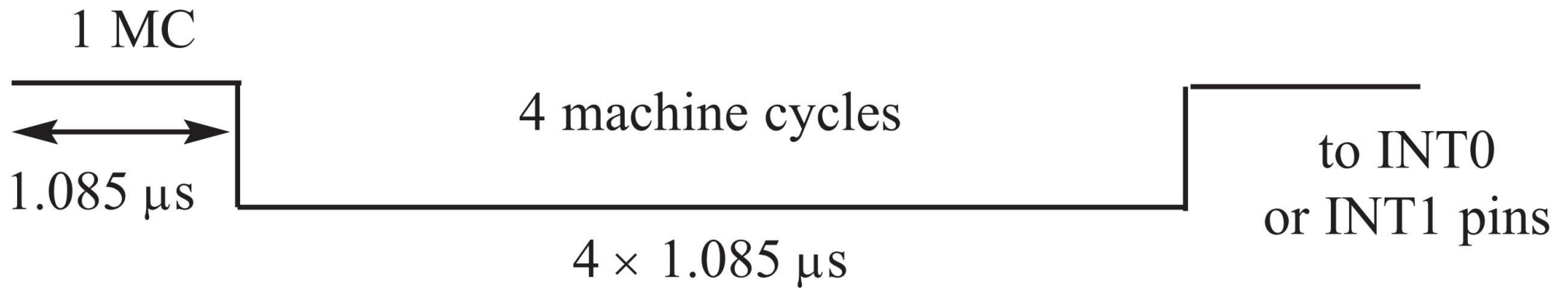


Figure 11-5. Minimum Duration of the Low Level-Triggered Interrupt (XTAL = 11.0592 MHz)



Note: On RESET, IT0 (TCON.0) and IT1 (TCON.2) are both low, making external interrupts level-triggered.

Figure 11-6. TCON (Timer/Counter) Register (Bit-addressable)

D7				D0			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TF1	TCON.7	Timer 1 overflow flag. Set by hardware when timer/counter 1 overflows. Cleared by hardware as the processor vectors to the interrupt service routine.					
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 on/off.					
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when timer/counter 0 overflows. Cleared by hardware as the processor vectors to the service routine.					
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 on/off.					
IE1	TCON.3	External interrupt 1 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed. Note: This flag does not latch low-level triggered interrupts.					
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt.					
IE0	TCON.1	External interrupt 0 edge flag. Set by CPU when external interrupt (H-to-L transition) edge is detected. Cleared by CPU when interrupt is processed. Note: This flag does not latch low-level triggered interrupts.					
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt.					

Example 11-6

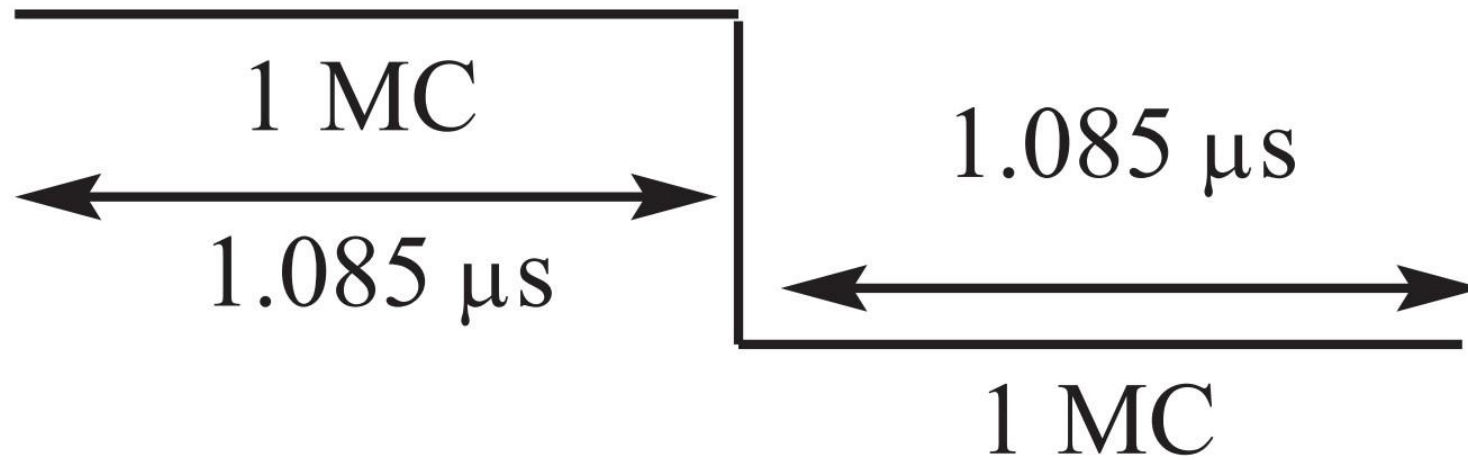
Assuming that pin 3.3 (INT1) is connected to a pulse generator, write a program in which the falling edge of the pulse will send a high to P1.3, which is connected to an LED (or buzzer). In other words, the LED is turned on and off at the same rate as the pulses are applied to the INT1 pin. This is an edge-triggered version of Example 11-5.

Solution:

```

        ORG      0000H
        LJMP     MAIN
;--ISR for hardware interrupt INT1 to turn on the LED
        ORG      0013H                      ;INT1 ISR
        SETB     P1.3                        ;turn on the LED
        MOV      R3,#255
BACK:    DJNZ     R3,BACK                    ;keep the LED on for a while
        CLR      P1.3                        ;turn off the LED
        RETI                                     ;return from ISR
;--MAIN program for initialization
        ORG      30H
MAIN:    SETB     TCON.2                    ;make INT1 edge-trigger interrupt
        MOV      IE,#10000100B            ;enable External INT1
HERE:    SJMP     HERE                      ;stay here until interrupted
        END
```

Minimum Pulse Duration to Detect Edge-triggered Interrupts. XTAL = 11.0592 MHz



Example 11-7

What is the difference between the RET and RETI instructions? Explain why we cannot use RET instead of RETI as the last instruction of an ISR.

Solution:

Both perform the same actions of popping off the top two bytes of the stack into the program counter, and making the 8051 return to where it left off. However, RETI also performs an additional task of clearing the interrupt-in-service flag, indicating that the servicing of the interrupt is over and the 8051 now can accept a new interrupt on that pin. If you use RET instead of RETI as the last instruction of the interrupt service routine, you simply block any new interrupt on that pin after the first interrupt, since the pin status would indicate that the interrupt is still being serviced. In the cases of TF0, TF1, TCON.1, and TCON.3, they are cleared by the execution of RETI.

Figure 11-7. Single Interrupt for Both TI and RI



Serial interrupt is invoked by TI or RI flags

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.

Example 11-9

Write a program in which the 8051 gets data from P1 and sends it to P2 continuously while incoming data from the serial port is sent to P0. Assume that XTAL = 11.0592 MHz. Set the baud rate at 9600.

Solution:

```

                                ORG      0
                                LJMP     MAIN
                                ORG      23H
                                LJMP     SERIAL          ;jump to serial 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      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
                                MOV      P0,A           ;send incoming 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
```

Table 11-2: Interrupt Flag Bits for the 8051/52

Interrupt	Flag	SFR Register Bit
External 0	IE0	TCON.1
External 1	IE1	TCON.3
Timer 0	TF0	TCON.5
Timer 1	TF1	TCON.7
Serial port	T1	SCON.1
Timer 2	TF2	T2CON.7 (AT89C52)
Timer 2	EXF2	T2CON.6 (AT89C52)

Example 11-10

Write a program using interrupts to do the following: (a) Receive data serially and send it to P0, (b) Have port P1 read and transmitted serially, and a copy given to P2, (c) Make Timer 0 generate a square wave of 5 kHz frequency on P0.1. Assume that XTAL = 11.0592 MHz. Set the baud rate at 4800.

Solution:

```

                                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 int. ISR
                                ORG      30H
MAIN:                            MOV     P1,#0FFH        ;make P1 an input port
                                MOV     TMOD,#22H        ;timer 0&1,mode 2, auto-reload
                                MOV     TH1,#0F6H        ;4800 baud rate
                                MOV     SCON,#50H        ;8-bit, 1 stop, REN enabled
                                MOV     TH0,#-92         ;for 5 KHz wave
                                MOV     IE,#10010010B    ;enable serial, timer 0 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            ;write it to P2
                                SJMP     BACK            ;stay in loop indefinitely
;-----SERIAL PORT ISR
SERIAL:                          ORG      100H
                                JB       TI,TRANS        ;jump if TI is high
                                MOV     A,SBUF           ;otherwise due to received
                                MOV     P0,A            ;send serial data to P0
                                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
```

Table 11-3: 8051/52 Interrupt Priority Upon Reset

Highest to Lowest Priority

External Interrupt 0	(INT0)
Timer Interrupt 0	(TF0)
External Interrupt 1	(INT1)
Timer Interrupt 1	(TF1)
Serial Communication	(RI + TI)
Timer 2 (8052 only)	TF2

Example 11-11

Discuss what happens if interrupts INT0, TF0, and INT1 are activated at the same time. Assume priority levels were set by the power-up reset and that the external hardware interrupts are edge-triggered.

Solution:

If these three interrupts are activated at the same time, they are latched and kept internally. Then the 8051 checks all five interrupts according to the sequence listed in Table 11-3. If any is activated, it services it in sequence. Therefore, when the above three interrupts are activated, IE0 (external interrupt 0) is serviced first, then Timer 0 (TF0), and finally IE1 (external interrupt 1).

Figure 11-8. Interrupt Priority Register (Bit-addressable)



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

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

User software should never write 1s to unimplemented bits, since they may be used in future products.

Example 11-12

(a) Program the IP register to assign the highest priority to INT1 (external interrupt 1), then (b) discuss what happens if INT0, INT1, and TF0 are activated at the same time. Assume that the interrupts are both edge-triggered.

Solution

(a) `MOV IP,#00000100B ;IP.2=1 to assign INT1 higher priority`

The instruction “SETB IP.2” also will do the same thing as the above line since IP is bit-addressable.

(b) The instruction in Step (a) assigned a higher priority to INT1 than the others; therefore, when INT0, INT1, and TF0 interrupts are activated at the same time, the 8051 services INT1 first, then it services INT0, then TF0. This is due to the fact that INT1 has a higher priority than the other two because of the instruction in Step (a). The instruction in Step (a) makes both the INT0 and TF0 bits in the IP register 0. As a result, the sequence in Table 11-3 is followed, which gives a higher priority to INT0 over TF0.

Example 11-13

Assume that after reset, the interrupt priority is set by the instruction “MOV IP, #00001100B”. Discuss the sequence in which the interrupts are serviced.

Solution

The instruction “MOV IP,#00001100B” (B is for binary) sets the external interrupt 1 (INT1) and Timer 1 (TF1) to a higher priority level compared with the rest of the interrupts. However, since they are polled according to Table 11-3, they will have the following priority.

Highest Priority	External Interrupt 1	(INT1)
	Timer Interrupt 1	(TF1)
	External Interrupt 0	(INT0)
	Timer Interrupt 0	(TF0)
Lowest Priority	Serial Communication	(RI + TI)

Table 11-4: 8051/52 Interrupt Numbers in C

Interrupt	Name	Numbers used by 8051 C
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4
Timer 2 (8052 only)	(TF2)	5