

Contents

- **Interrupt Programming**

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
 - 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*

INTERRUPTS

Steps in Executing an Interrupt

- ❑ 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 location in memory, called the interrupt vector table, that holds the address of the ISR

INTERRUPTS

Steps in Executing an Interrupt (cont')

(cont')

4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it
 - It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt)
5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted
 - First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC
 - Then it starts to execute from that address

RET & RETI Instructions of 8051 μ C

RET	RETI
❖ Used with normal subroutine.	❖ Used with ISR – Interrupt Service Routine
❖ With RET, 8051 just return back to main program from subroutine.	❖ With RETI, 8051 return back to main program + It will Enable Interrupt by making EA = 1.
❖ Operation: POP PC ; PCH $\leftarrow$ [SP] ; PCL $\leftarrow$ [SP-1] ; SP $\leftarrow$ SP - 2	❖ Operation: POP PC ; PCH $\leftarrow$ [SP] ; PCL $\leftarrow$ [SP-1] ; SP $\leftarrow$ SP - 2 EA $\leftarrow$ 1

Six Interrupts in 8051:

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.

INTERRUPTS

Enabling and Disabling an Interrupt (cont')

IE (Interrupt Enable) Register

D7

D0

EA

--

ET2

ES

ET1

EX1

ET0

EX0

EA (enable all) must be set to 1 in order for rest of the register to take effect

EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
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

Interrupt Requests Enable Via IE register

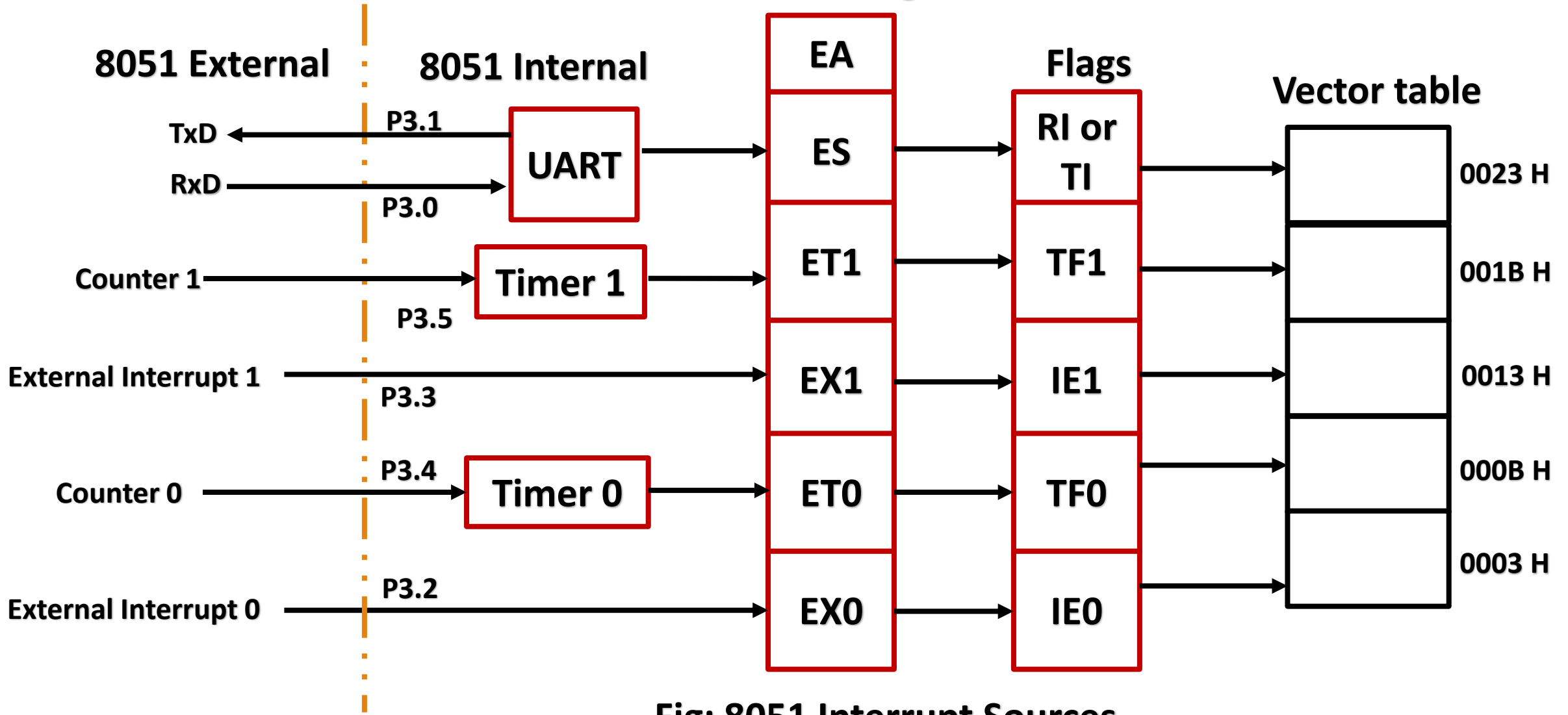


Fig: 8051 Interrupt Sources

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`

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`

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

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

• • •

[illegible]

Example 11-3

Rewrite Example 11-2 to create a square wave that has a high portion of 1085 us and a low portion of 15 us. Assume XTAL=11.0592MHz. Use timer 1.

Solution:

Since 1085 us is 1000×1.085 we need to use mode 1 of timer 1.

;--upon wake-up go to main, avoid using
;memory allocated to Interrupt Vector Table

```
ORG 0000H
```

```
LJMP MAIN      ;by-pass int. vector table
```

;--ISR for timer 1 to generate square wave

```
ORG 001BH      ;Timer 1 int. vector table
```

```
LJMP ISR_T1     ;jump to ISR
```

...

...

;--The main program for initialization

```
ORG 0030H      ;after vector table space
```

```
MAIN: MOV TMOD,#10H ;Timer 1, mode 1
```

```
MOV P0,#0FFH   ;make P0 an input port
```

```
MOV TL1,#018H  ;TL1=18 low byte of -1000
```

```
MOV TH1,#0FCH  ;TH1=FC high byte of -1000
```

```
MOV IE,#88H    ;10001000 enable Timer 1 int
```

```
SETB TR1       ;Start Timer 1
```

```
BACK: MOV A,P0  ;get data
```

```
MOV P1,A       ;issue
```

```
SJMP BACK      ;keep
```

;Timer 1 ISR. Must be reloaded, not auto-reload

```
ISR_T1: CLR TR1 ;stop Timer 1
```

```
MOV R2,#4      ;
```

```
CLR P2.1       ;P2.1=0, start of low portion
```

```
HERE: DJNZ R2,HERE ;4x2 machine cycle
```

```
MOV TL1,#18H   ;load T1 low byte value
```

```
MOV TH1,#0FCH  ;load T1 high byte value
```

```
SETB TR1       ;starts timer1
```

```
SETB P2.1      ;P2.1=1, back to high
```

```
RETI           ;return to main
```

```
END
```

Low portion of the pulse is
created by 14 MC
 $14 \times 1.085 \text{ us} = 15.19 \text{ us}$

2MC

8MC

2MC

2MC

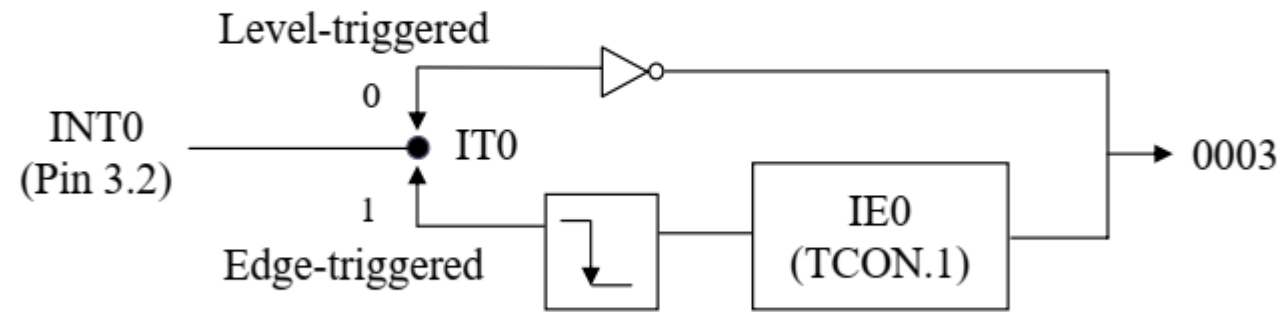
1MC

1MC

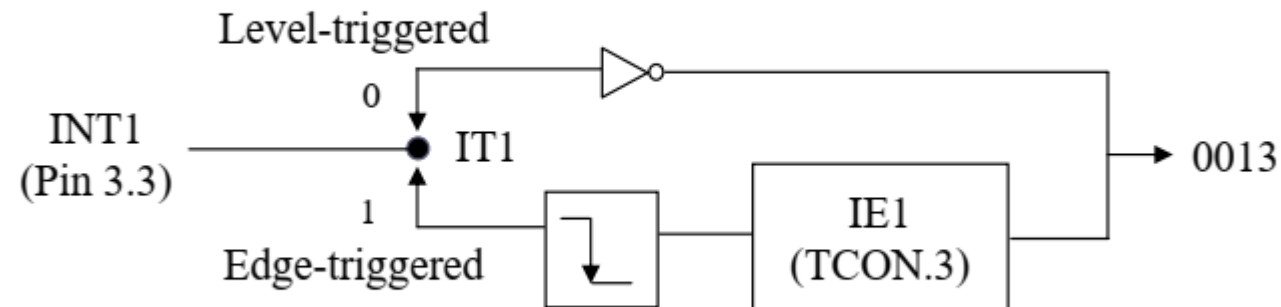
EXTERNAL HARDWARE INTERRUPTS

- ❑ The 8051 has two external hardware interrupts
 - Pin 12 (P3.2) and pin 13 (P3.3) of the 8051, designated as INT0 and INT1, are used as external hardware interrupts
 - 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
 - Level triggered
 - Edge triggered

Activation of INT0



Activation of INT1



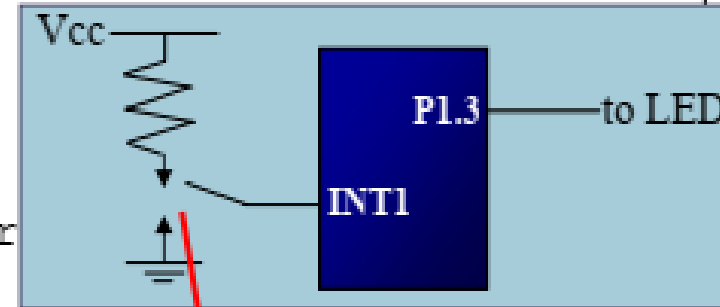
Example 11-5

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 ;by-pass interrupt vector table
;--ISR for INT1 to turn on LED
ORG 0013H ;INT1 ISR
SETB P1.3 ;turn on LED
MOV R3,#255
BACK: DJNZ R3,BACK ;keep LED on for a fraction of a second
CLR P1.3 ;turn off the LED
RETI ;return from ISR

;--MAIN program for initialization
ORG 30H
MAIN: MOV IE,#10000100B ;enable external INT 1
HERE: SJMP HERE ;stay here until get interrupted
END
```



Pressing the switch will cause the LED to be turned on. If it is kept activated, the LED stays on

- In previous problem , to ensure the activation of hardware interrupt at the INT1 pin, make sure that the duration of the low-level signal is around 4 machine cycles, but no more.
- Upon reset the 8051 makes INT0 and INT1 as low-level triggered interrupts.

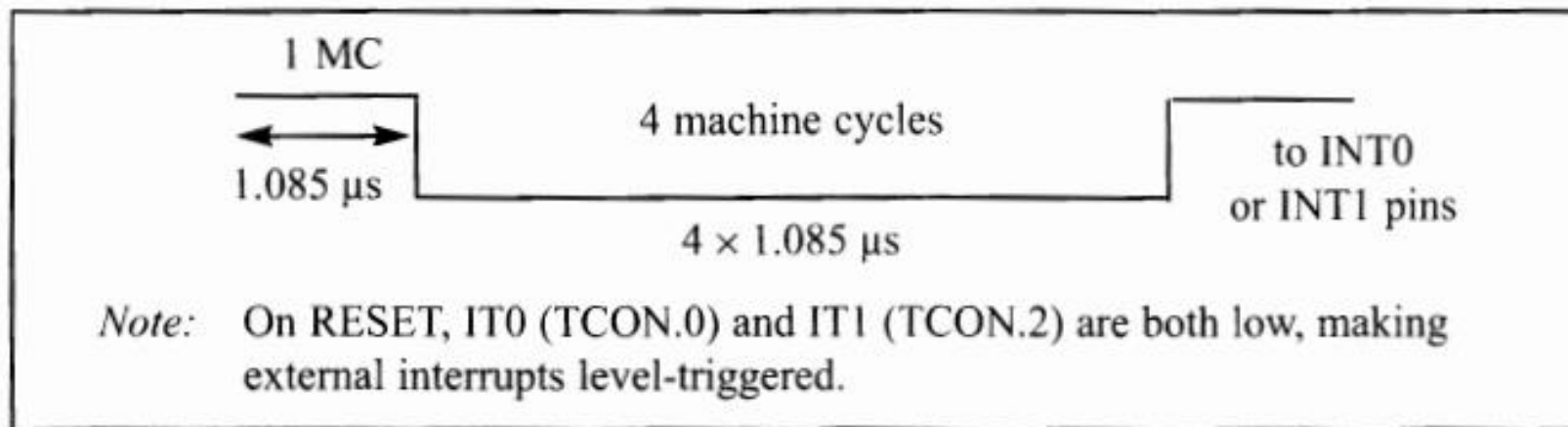


Figure 11-5. Minimum Duration of the Low Level-Triggered Interrupt (XTAL = 11.0592 MHz)

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. <i>Note:</i> 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. <i>Note:</i> 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.					

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

EXTERNAL HARDWARE INTERRUPTS

Edge-Triggered Interrupt (cont')

The on-state duration depends on the time delay inside the ISR for INT1

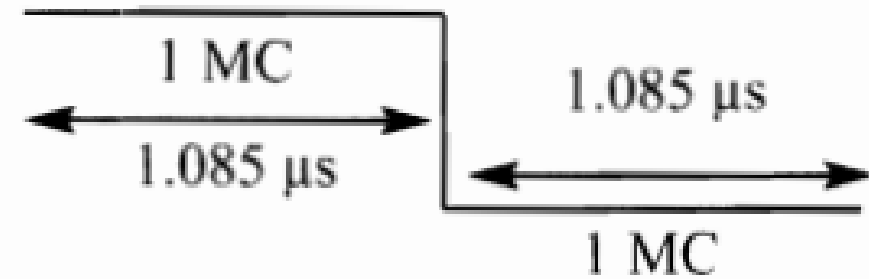
Assume that pin 3.3 (INT1) is connected to a pulse generator, write a program in which the falling edge of the pulse will send a high to P1.3, which is connected to an LED (or buzzer). In other words, the LED is turned on and off at the same rate as the pulses are applied to the INT1 pin.

Solution:

```
ORG 0000H
LJMP MAIN
;--ISR for hardware interrupt INT1 to turn on LED
ORG 0013H ;INT1 ISR
SETB P1.3 ;turn on LED
MOV R3,#255
BACK: DJNZ R3,BACK ;keep the buzzer on for a while
CLR P1.3 ;turn off the buzzer
RETI ;return from ISR
;-----MAIN program for initialization
ORG 30H
MAIN: SETB TCON.2 ;make INT1 edge-triggered int.
MOV IE,#10000100B ;enable External INT 1
HERE: SJMP HERE ;stay here until get interrupted
END
```

When the falling edge of the signal is applied to pin INT1, the LED will be turned on momentarily.

Minimum pulse duration to detect
edge-triggered interrupts.
XTAL = 11.0592 MHz



SERIAL COMMUNI- CATION INTERRUPT

Use of Serial COM in 8051 (cont')

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. Set the baud rate at 9600.

Solution:

```
ORG 0000H
LJMP MAIN
ORG 23H
LJMP SERIAL ;jump to serial int ISR
ORG 30H
MAIN: MOV P1,#0FFH ;make P1 an input port
      MOV TMOD,#20H ;timer 1, auto reload
      MOV TH1,#0FDH ;9600 baud rate
      MOV SCON,#50H ;8-bit, 1 stop, ren enabled
      MOV IE,10010000B ;enable serial int.
      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 COMMUNI- CATION INTERRUPT

Use of Serial COM in 8051 (cont')

```
...  
;-----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 doesn't  
                RETI        ;return from ISR  
TRANS:  CLR TI      ;clear TI since CPU doesn't  
                RETI        ;return from ISR  
                END
```

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, and that 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 interrupt.

SERIAL COMMUNI- CATION INTERRUPT

Use of Serial COM in 8051 (cont')

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. Set the baud rate at 9600.

Solution:

```
ORG 0000H
LJMP MAIN
ORG 23H
LJMP SERIAL ;jump to serial int ISR
ORG 30H
MAIN: MOV P1,#0FFH ;make P1 an input port
      MOV TMOD,#20H ;timer 1, auto reload
      MOV TH1,#0FDH ;9600 baud rate
      MOV SCON,#50H ;8-bit, 1 stop, ren enabled
      MOV IE,10010000B ;enable serial int.
      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 COMMUNI- CATION INTERRUPT

Use of Serial
COM in 8051
(cont')

```
...  
;-----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
```

SERIAL COMMUNI- CATION INTERRUPT

Clearing RI and
TI before RETI

Example 11-10

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.

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 interrupt ISR
ORG 30H
MAIN: MOV P1,#0FFH ;make P1 an input 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
...
```

SERIAL COMMUNI- CATION INTERRUPT

Clearing RI and
TI before RETI
(cont')

```
...      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
;-----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 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
```

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

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

INTERRUPT PRIORITY (cont')

Interrupt Priority Register (Bit-addressable)

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

--	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

Priority bit=1 assigns high priority

Priority bit=0 assigns low priority

INTERRUPT PRIORITY (cont')

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 the interrupts are both edge-triggered.

Solution:

- (a) `MOV IP, #00000100B ; IP.2=1` 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

INTERRUPT PRIORITY (cont')

Example 11-13

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

Solution:

The instruction “MOV IP #00001100B” (B is for binary) and timer 1 (TF1) to a higher priority level compared with the reset of the interrupts. However, since they are polled according to Table, 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)

Thank You