8051 Interrupts

Interrupt & Polling

- A single microcontroller always connects to serve several peripheral devices through its I/O ports
- There are two ways for the peripheral devices to request service from microcontroller
  - Polling
  - Interrupt

Programmed I/O (Polling)

- Microcontroller continuously monitors the status of a certain number of devices in sequence
- Services to a device if preset condition met
- After the service, the microcontroller will move on to monitor the status of another device until all devices are serviced
- The operation described above is called "**polling**"

Interrupt I/O (Interrupt)

- Whenever any device needs the service, it notifies the microcontroller by sending it an **interrupt request** (**IR**) signal while the microcontroller is doing other work.
- The microcontroller suspends its work to service the device at once.
- Note that each IR is associated with an **interrupt service routine** (**ISR**)

Comparison between Interrupt and Polling

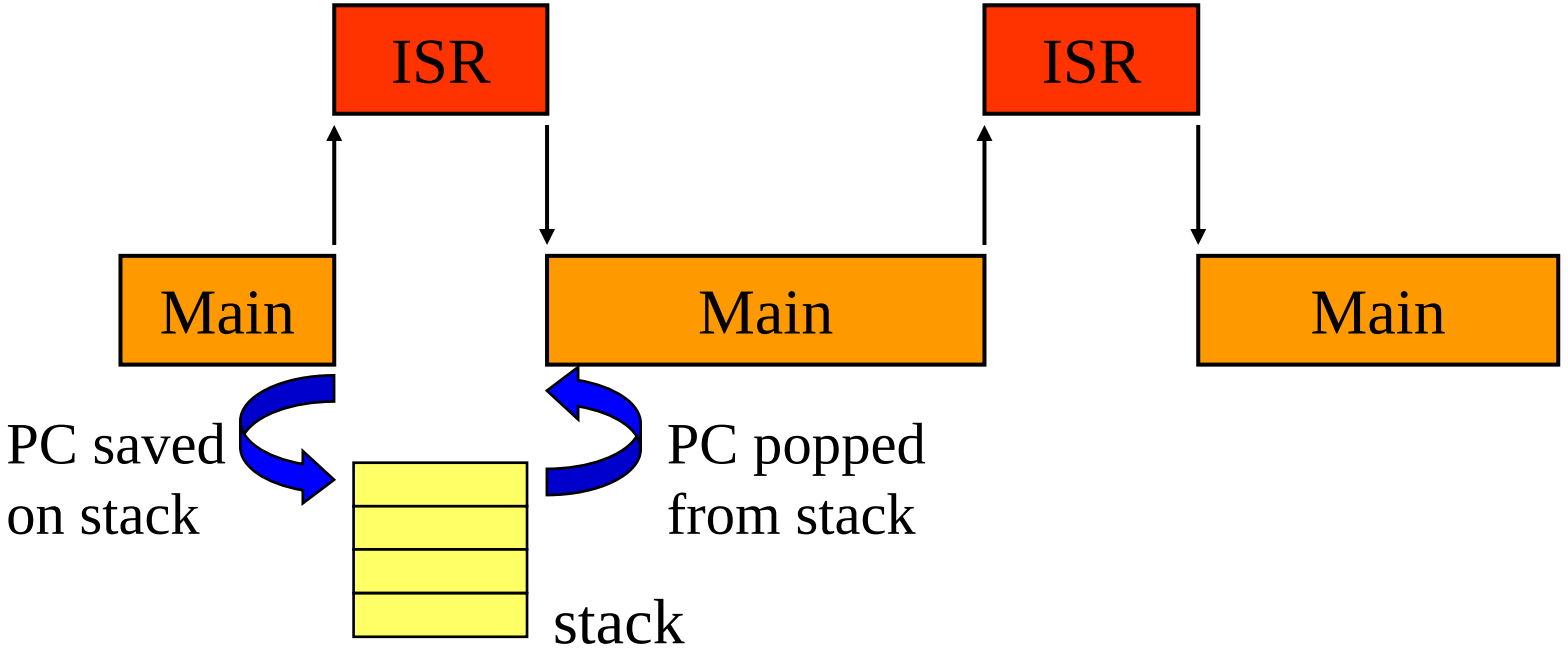
	Interrupt	Polling
Method	Devices notify MCU by sending it an interrupt signals while the MCU is doing another work	MCU continuously monitors devices to determine whether they need service
Response time	Faster	Slower
Need of MCU time	Less	More
Priority setting	Yes	No

Steps in Handling an Interrupt Request

When an interrupt activates and is accepted by the MCU, the main program is interrupted. The following actions occurs:

- The current instruction will be finished
- The PC is saved on the stack
- The current interrupt status is saved internally
- The PC is loaded with the vector address of the ISR from the interrupt vector table (Jump to execute ISR)
- The ISR is executed and will be finished with a RETI instruction (return from interrupt)
- Return to main program by popping the PC from the stack

Program Execution with Interrupts



The program that deals with an interrupt is called an **interrupt service routine** (**ISR**)

ISR executes in response to the interrupt and generally performs I/O operation to a device

Types of Interrupt in the 8051

There are 6 interrupts in the 8051

- Reset – when the reset pin is activated, the 8051 will reset all registers and ports and jumps to address location 0000H starting up execution.
- 2 external interrupts – Hardware external interrupts (INT0 and INT1) at pins 12 & 13 are used to receive interrupt signals from external devices.
- 2 timer interrupts – They are Timer 0 and Timer 1 which will give out interrupt signal when the timers count to zero.
- 1 serial port interrupt - It is used for data reception and transmission during serial communication.

Apart from the Reset, only the 5 interrupts are available to the user

Interrupt Vector Table for the 8051

- **interrupt vector table** holds the addresses of ISR

Priority	Interrupt	Flag	ROM location	Pin
1	Reset	RST	0000H	9
2	External 0 (INT0)	IE0	0003H	P3.2 (12)
3	Timer 0	TF0	000BH	---
4	External 1 (INT1)	IE1	0013H	P3.3 (13)
5	Timer 1	TF1	001BH	---
6	Serial port	RI or TI	0023H	---

Enabling and Disabling an Interrupt

- Upon reset, all interrupts are disabled
- The interrupts must be enabled by software
- A register called **IE (interrupt enable)** register, which is bit-addressable, is responsible for enabling and disabling the interrupts
- Bit IE. 7 must be set high to allow the rest of register to take effect
  - EA = 1 ; Global enable interrupt
  - EA = 0 ; Global disable interrupt

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

IE (interrupt enable) register

Enabling and Disabling an Interrupt

Bit	Symbol	Description (1=Enable, 0=Disable)
IE.7	EA	Global Enable/disable
IE.6	--	Not implemented, reserved for future use
IE.5	ET2	Not use for 8051 (8052 only)
IE.4	ES	Enable/disable serial port interrupt
IE.3	ET1	Enable/disable timer 1 interrupt
IE.2	EX1	Enable/disable external interrupt 1
IE.1	ET0	Enable/disable timer 0 interrupt
IE.0	EX0	Enable/disable external interrupt 0

Example 1

Write instructions to

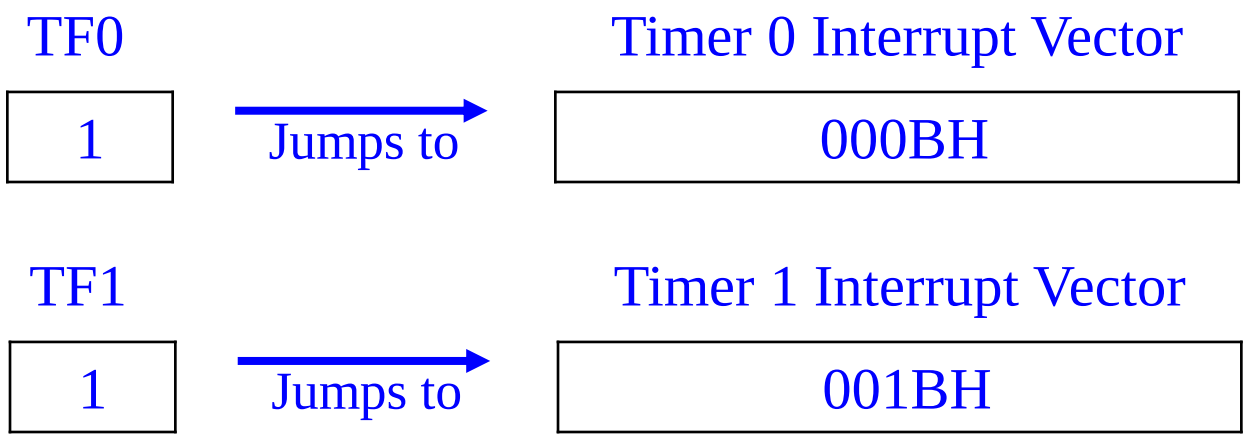
(a) Enable serial interrupt, Timer 0 interrupt and external interrupt 1, and
(b) Disable Timer 0 interrupt only, then
(c) Disable all the interrupt with a single instruction

Solution:

```
(a)     MOV  IE, #10010110B ; enable serial, Timer 0, EX1 interrupts
    or
        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  interrupt
(b)     CLR  IE.1           ; disable Timer 0 interrupt
(c)     CLR  IE.7           ;  disable all interrupts
```

Timer Interrupt

- When timer rolls over, its timer flag (TF) is set.
- If the timer interrupt in the IE register is enable, whenever the TF is set, the microcontroller is interrupted and jumps to the interrupt vector table to service the ISR.
- With timer interrupt is enabled, microcontroller can do other things and no need to monitor the TF for rolling over.



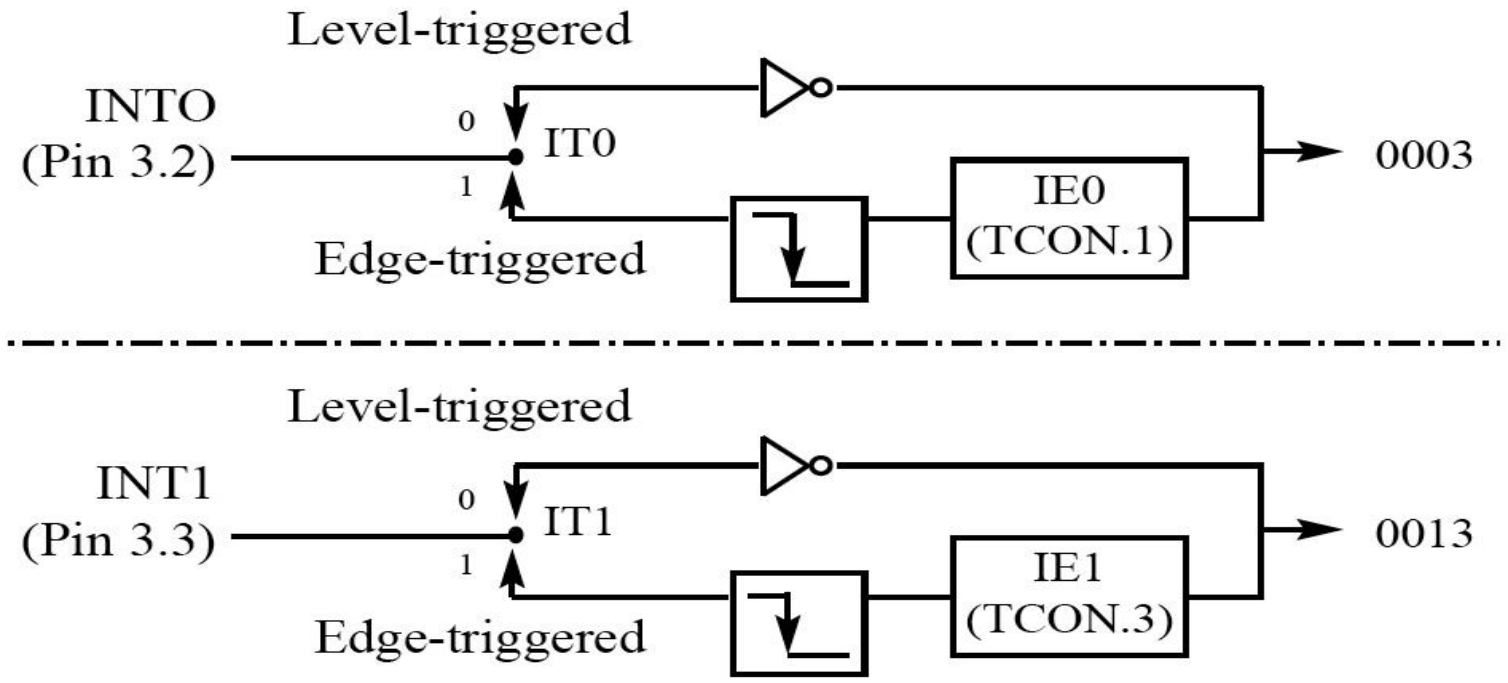
Example 2

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.

```
01  ORG 0000H
02  LJMP MAIN                   ;bypass interrupt vector table
03
04  ;ISR for Timer 0 to generate square wave
05  ORG 000BH                   ;Timer 0 interrupt vector table
06  CPL P2.1                    ;toggle P2.1 pin
07  RETI
08
09  ;The main program for initialization
10  ORG 0030H                   ;after vector table space
11  MAIN: MOV TMOD,#02H         ;Timer 0, mode 2(auto-reload)
12  MOV P0,#0FFH                ;make P0 an input port
13  MOV TH0,#-92                ;TH0=A4H for -92
14  MOV IE,#82H                 ;IE=10000010(bin) enable Timer 0
15  SETB TR0                    ;Start Timer 0
16
17  BACK: MOV A,P0              ;get data from P0
18  MOV P1,A                    ;issue it to P1
19  SJMP BACK                   ;keep doing it
20
21  END
```

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

- **External interrupts INT0 and INT1**



Activation of INT0 and INT1

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

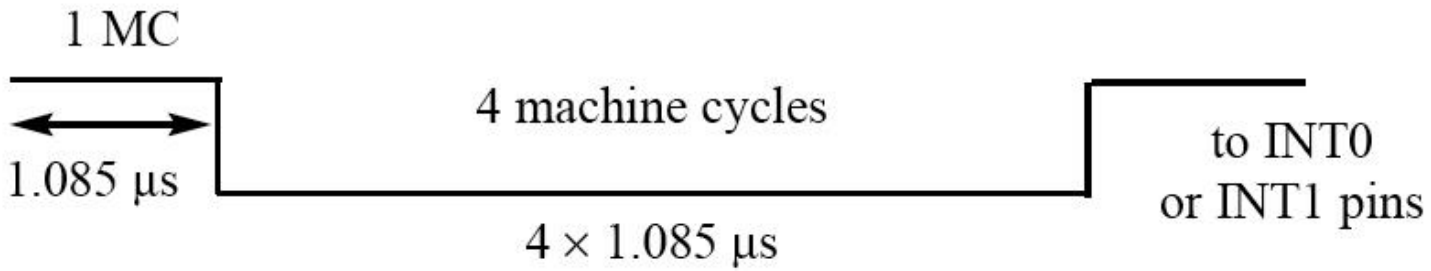
- **Level-triggered interrupt**
  - **INT0 and INT1 pins are normally high**
  - **if low-level signal is applied, it triggers the interrupt**
  - **microcontroller stops whatever it is doing and jumps to the interrupt vector table to service the interrupt**
  - **the lowlevel signal must be removed before the execution of the last instruction of the interrupt service routine, RETI**
  - **otherwise, another interrupt will be generated**

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

- Sampling the low level-triggered interrupt
  - to ensure the activation of the hardware interrupt at the INTx pin, make sure that the duration of the low-level signal is around 4 machine cycles

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

- **Sampling the low level-triggered interrupt**



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

Minimum Duration of the Low Level-Triggered Interrupt (XTAL = 11.0592 MHz)

- # Edge-triggered interrupts

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.

TCON (Timer/Counter) Register (Bit-addressable)

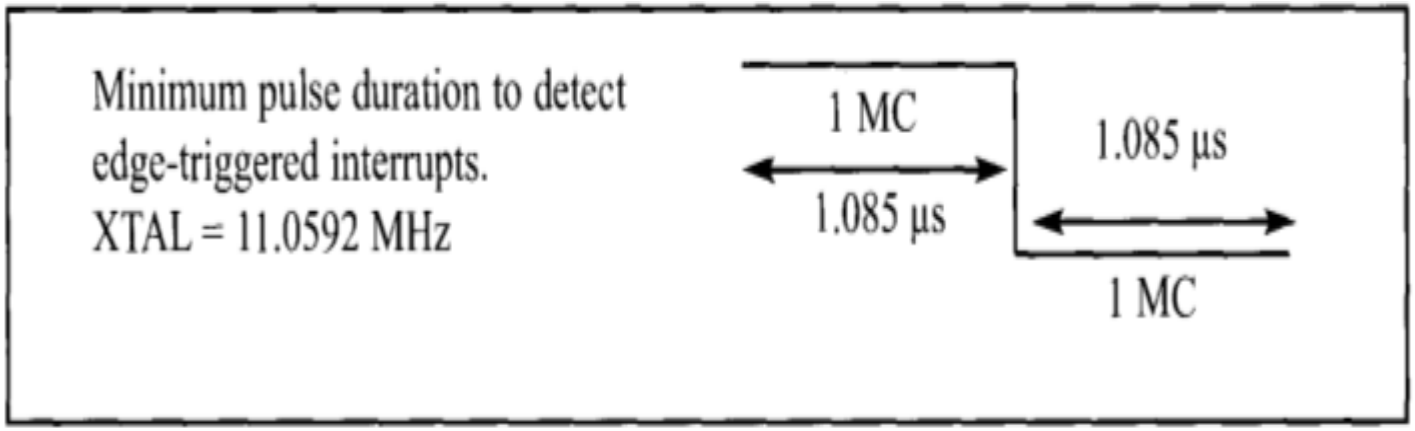
Example 6

Assuming that INT1 is connected to a pulse generator. Write a program in which the falling edge of the pulse will send a high to P 1.3, which is connected to an LED.

```
01  ORG 0000H
02  LJMP MAIN
03
04  ;ISR for hardware interrupt INT1 to turn on the LED
05  ORG 0013H             ;INT1 ISR
06  SETB P1.3             ;turn on the LED
07  MOV R3,#255
08  BACK:DJNZ R3,BACK     ;keep the LED on for a while
09  CLR P1.3              ;turn off the LED
10  RETI                  ;return from ISR
11
12  ;MAIN program for initialization
13  ORG 30H
14  MAIN: SETB TCON.2     ;make INT1 edge-trigger interrupt
15  MOV IE,#10000100B     ;enable External INT1
16  HERE: SJMP HERE       ;stay here until interrupted
17
18  END
```

SECTION 11.3: PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

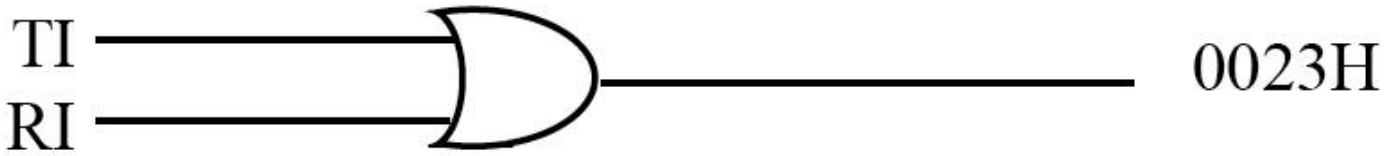
- Sampling the edge-triggered interrupt
  - external source must be held high for at least one machine cycle, and then held low for at least one machine cycle to ensure that the transition is seen by the microcontroller



SECTION 11.4: PROGRAMMING THE SERIAL COMMUNICATION INTERRUPT

- RI and TI flags and interrupts
  - 1 interrupt is set for serial communication
  - used to both send and receive data
  - when RI or TI is raised the 8051 gets interrupted and jumps to memory address location 0023H to execute the ISR
  - the ISR must examine the TI and RI flags to see which one caused the interrupt and respond accordingly

SECTION 11.4: PROGRAMMING THE SERIAL COMMUNICATION INTERRUPT



Serial interrupt is invoked by TI or RI flags

Single Interrupt for Both TI and RI

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.

```
01  ORG 0
02  LJMP MAIN
03  ORG 23H
04  LJMP SERIAL                ;jump to serial interrupt ISR
05  ORG 30H
06  MAIN: MOV P1,#0FFH         ;make P1 an input port
07  MOV TMOD,#20H              ;timer 1, mode 2(auto-reload)
08  MOV TH1,#0FDH              ;9600 baud rate
09  MOV SCON,#50H              ;8-bit, 1 stop, REN enabled
10  MOV IE,#10010000B          ;enable serial interrupt
11  SETB TR1                   ;start timer 1
12  BACK: MOV A,P1             ;read data from port 1
13  MOV SBUF,A                 ;give a copy to SBUF
14  MOV P2,A                   ;send it to P2
15  SJMP BACK                  ;stay in loop indefinitely
16  ;Serial Port ISR
17  ORG 100H
18  SERIAL:JB TI,TRANS         ;jump if TI is high
19  MOV A,SBUF                 ;otherwise due to receive
20  CLR RI                     ;clear RI since CPU does not
21  RETI                       ;return from ISR
22  TRANS: CLR TI              ;clear TI since CPU does not
23  RETI                       ;return from ISR
24  END
```

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.

```
01  ORG 0
02  LJMP MAIN
03  ORG 23H
04  LJMP SERIAL                ;jump to serial ISR
05  ORG 30H
06  MAIN: MOV P1,#0FFH         ;make P1 an input port
07  MOV TMOD,#20H              ;timer 1, mode 2(auto-reload)
08  MOV TH1,#0FDH              ;9600 baud rate
09  MOV SCON,#50H              ;8-bit,1 stop, REN enabled
10  MOV IE,#10010000B          ;enable serial interrupt
11  SETB TR1                   ;start Timer 1
12  BACK: MOV A,P1             ;read data from port 1
13  MOV P2,A                   ;send it to P2
14  SJMP BACK                  ;stay in loop indefinitely
15  ;SERIAL PORT ISR
16  ORG 100H
17  SERIAL: JB TI,TRANS        ;jump if TI is high
18  MOV A,SBUF                 ;otherwise due to receive
19  MOV P0,A                   ;send incoming data to P0
20  CLR RI            ;clear RI since CPU doesn't
21  RETI                       ;return from ISR
22  TRANS: CLR TI                       ;clear TI since CPU doesn't
23  RETI                       ;return from ISR
24  END
```

SECTION 11.4: PROGRAMMING THE SERIAL COMMUNICATION INTERRUPT

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)

Interrupt Flag Bits for the 8051/52

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.

```
01  ORG 0
02  LJMP MAIN
03  ORG 0BH                 ;ISR for Timer 0
04  CPL P0.1                ;toggle P0.1
05  RETI                    ;return from ISR
06  ORG 23H
07  LJMP SERIAL             ;jump to serial int.
08  ORG 30H
09  MAIN: MOV P1,#0FFH      ;make P1 an input port
10  MOV TMOD,#22H           ;timer 0&1,mode 2, auto-reload
11  MOV TH1,#0F6H           ;4800 baud rate
12  MOV SCON,#50H           ;8-bit, 1 stop, REN enabled
13  MOV TH0,#-92            ;for 5 KHz wave
14  MOV IE,#10010010B       ;enable serial, timer 0 int
15  ;SERIAL PORT ISR
16  ORG 100H
17  SERIAL: JB TI,TRANS     ;jump if TI is high
18  MOV A,SBUF              ;otherwise due to received
19  MOV P0,A                ;send serial data to P0
20  CLR RI                  ;clear RI since CPU does not
21  RETI                    ;return from ISR
22  TRANS: CLR TI           ;clear TI since CPU does not
23  RETI                    ;return from ISR
24  END
```

SECTION 11.5: INTERRUPT PRIORITY IN THE 8051/52

INT0 > TF0 > INT1 > TF1 > SERIAL(RI+TI)

- 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**

Table 11–3 8051/52 Interrupt Priority Upon Reset

SECTION 11.5: INTERRUPT PRIORITY IN THE 8051/52

- **Setting interrupt priority with the IP register**

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

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.

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

Setting Interrupt Priority with the IP register

Example 8.7 :

(a) Program the IP register to assign the highest priority to INT1,

(b) Discuss what happens if INT0, INT1 and TF0 are activated at the same time.

Solution :

(a) MOV IP, #00000100B ; set IP.2=1 INT1 has the highest priority

or SETB IP.2

(b) Priority of interrupt will be changed to INT1 > INT0 > TF0

The 8051 will services INT1 first and then INT0 and TF0.

(As the INT0 and TF0 bits in IP register are 0, their priorities follow the sequence in Table 8.1)

Setting Interrupt Priority with the IP register

Example 8.8 :

The IP register is set by the instruction “MOV IP, #00001100B” after reset. Discuss the sequence in which the interrupts are serviced.

Solution :

MOV IP, #00001100B instruction sets INT1 & TF1 to a higher priority level compared with the rest of the interrupts.

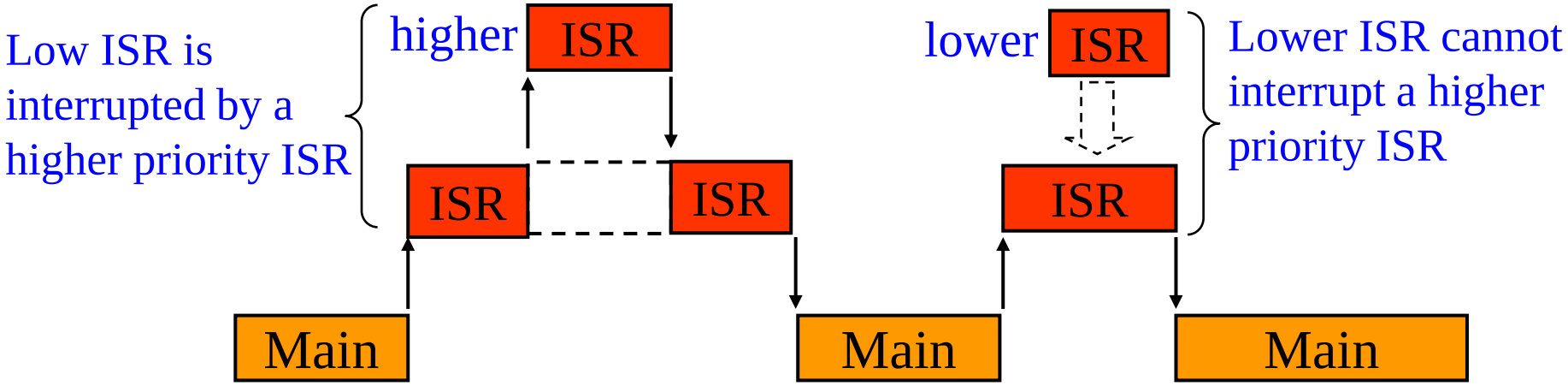
Priority of interrupt will then be

INT1 > TF1 > INT0 > TF0 > Serial

POWER UP DEFAULT PRIORITY: INT0 > TF0 > INT1 > TF1 > SERIAL(RI+TI)

Interrupt inside an interrupt

- Higher-priority interrupt can interrupt a low-priority interrupt.
- An interrupt cannot be interrupted by a low-priority interrupt
- No low priority interrupt can get the immediate attention of the CPU until the 8051 has finished servicing the high-priority interrupts

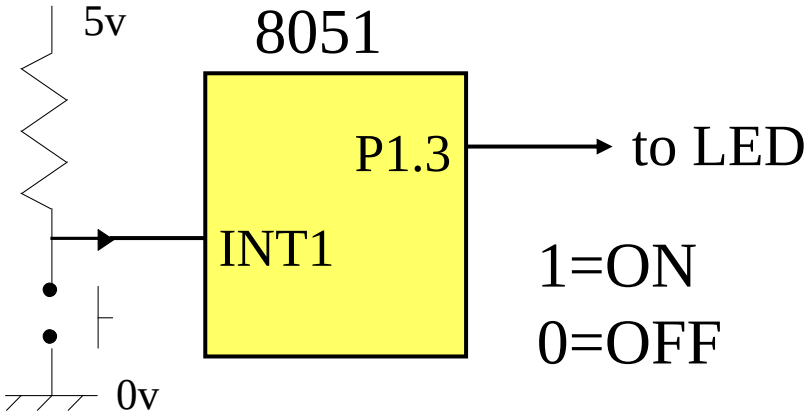


SECTION 11.5: INTERRUPT PRIORITY IN THE 8051/52

- Triggering the interrupt by software
  - can test an ISR with instructions to set the interrupts high
  - "SETB TF1" will interrupt the 8051 in whatever it is doing and force it to jump to the interrupt vector table
  - don’t have to wait for Timer 1 to roll over
  - useful for testing ISR

Example 8.9

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, the LED should stay on.



Example 8.9 - Solution

```
                ORG     0000H
                LJMP    MAIN                ;by-pass interrupt vector table

; ISR for INT1
                ORG     0013H
                LJMP    ISR_INT1            ; jump to ISR_INT1

; Main Program
                ORG     30H
MAIN:           MOV     IE,#10000100B       ;enable external INT 1 + GLOBAL
HERE:           SJMP    HERE                ;stay here until get interrupted

; INT1 ISR
ISR_INT1:       SETB    P1.3                ;turn on LED
                MOV     R3,#255
BACK:           DJNZ    R3, BACK            ;keep LED on for a while
                CLR     P1.3                ;turn off LED
                RETI
                END
```

🕮 Read reference

- The 8051 Microcontroller and Embedded Systems - Using Assembly and C, Mazidi
  - Chapter 9 P.239 – P.255
  - Chapter 11 P.317 – P.339