5.2 SERIAL PORT PROGRAMMING

Data transfer between the PC and an 8051 system without any error is possible, if the baud rate of the 8051 system matches the baud rate of the PC's COM port.

BAUD RATE IN THE 8051

The 8051 transfers and receives data serially at many different baud rates. Serial communications of the 8051 is established with PC through the COM port. It must make sure that the baud rate of the 8051 system matches the baud rate of the PC's COM port/ any system to be interfaced. The baud rate in the 8051 is programmable. This is done with the help of Timer. When used for serial port, the frequency of timer is determined by (XTAL/12)/32 and 1 bit is transmitted for each timer period.

The Relationship between the crystal frequency and the baud rate in the 8051 is that the 8051 divides the crystal frequency by 12 to get the machine cycle frequency which is shown in Figure 5.2.1. Here the oscillator is XTAL = 11.0592 MHz, the machine cycle frequency is 921.6 kHz. 8051's UART divides the machine cycle frequency of 921.6 kHz by 32 once more before it is used by Timer 1 to set the baud rate. 921.6 kHz divided by 32 gives 28,800 Hz. Timer 1 in mode 2 is used to set the baud rate.

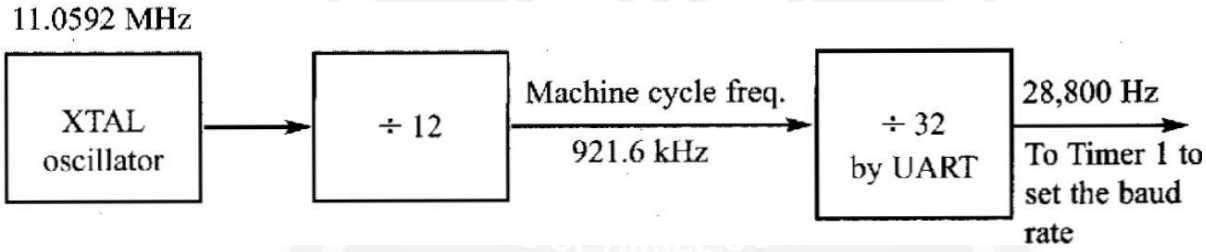


Figure 5.2.1 Frequency required to set the Baud rate

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.288]

CALCULATION OF BAUD RATE:

In serial communication if data transferred with a baud rate of 9600 and XTAL used is 11.0592 MHz, then following steps to be followed to find the TH1 value to be loaded.

Clock frequency of timer clock: f = (11.0592 MHz / 12)/32 = 28,800Hz

Time period of each clock tick: T0 = 1/f = 1/28800

Duration of timer : n*T0 (n is the number of clock ticks)

9600 baud ->duration of 1 symbol: 1/9600

1/9600 = n*T0 = n*1/28800

n = f/9600 = 28800/9600 = 3 ->TH1 =-3

Similarly, for baud 2400

n = f/2400 = 12 ->TH1 = -12

BAUD RATE SELECTION

Baud rate is selected by timer1 and when Timer 1 is used to set the baud rate it must be programmed in mode 2 that is 8-bit, auto-reload. To get baud rates compatible with the PC, we must load TH1 with the values shown in Table 5.2.1.

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	E8

Note: XTAL = 11.0592 MHz.

Table 5.2.1 Timer 1 THI register values for different baud rates

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.287]

REGISTERS FOR SERIAL COMMUNICATION

SBUF (SERIAL BUFFER) REGISTER:

It is an 8 bit register used solely for serial communication in the 8051.A byte of data to be transferred via the TxD line must be placed in the SBUF register. SBUF holds the byte of data when it is received by the RxD line. It can be accessed like any other register.

When a byte is written, it is framed with the start and stop bits and transferred serially via the TxD pin and when the bits are received serially via RxD, it is deframed

by eliminating the stop and start bits, making a byte out of the data received, and then placing it in the SBUF.

SCON (SERIAL CONTROL) REGISTER:

It is an 8 bit register used to program start bit, stop bit, and data bits of data framing, among other things.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication. (Make it 0.)
REN	SCON.4	Set/cleared by software to enable/disable reception.
TB8	SCON.3	Not widely used.
RB8	SCON.2	Not widely used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the beginning of the stop bit in mode 1. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware halfway through the stop bit time in mode 1. Must be cleared by software.

Note: Make SM2, TB8, and RB8 = 0.

Figure 5.2.2 SCON Register

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.289]

STEPS TO SEND DATA SERIALLY:

1.Set baud rate by loading TMOD register with the value 20H, this indicating timer 1 in mode 2 (8-bit auto-reload) to set baud rate

2. The TH1 is loaded with proper values to set baud rate for serial data transfer

3. The SCON register is loaded with the value 50H, indicating serial 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. TI is cleared by CLR TI instruction

6. The character byte to be transferred serially is written into SBUF register

7. The TI flag bit is monitored with the use of instruction JNB TI,xx to see if the character has been transferred completely

8. To transfer the next byte, go to step 5

Program to transfer letter "A" serially at 9800baud, continuously:

```
MOV TMOD,#20H          ; timer 1,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 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
```

IMPORTANCE OF THE TI FLAG:

Check the TI flag bit, we know whether or not 8051 is ready to transfer another byte. TI flag bit is raised by the 8051 after transfer of data. TI flag is cleared by the programmer by instruction like "CLR TI". When writing a byte into SBUF, before the TI flag bit is raised, it may lead to loss of a portion of the byte being transferred

STEPS TO RECEIVE DATA SERIALLY:

1. Set baud rate by loading TMOD register with the value 20H, this indicating timer 1 in mode 2 (8-bit auto-reload) to set baud rate .
2. The TH1 is loaded with proper values to set baud rate
3. The SCON register is loaded with the value 50H, indicating serial 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. RI is cleared by CLR RI instruction
6. The RI flag bit is monitored with the use of instruction JNB RI,xx to see if an entire character has been received yet
7. When RI is raised, SBUF has the byte; its contents are moved into a safe place
8. To receive next character, go to step 5

Program to receive bytes of data serially, and put them in P2, set the baud rate at 9600, 8-bit data, and 1 stop bit:

```
MOV TMOD, #20H          ; timer 1,mode 2(auto reload)
MOV TH1, #-3            ; 9600 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 P2, A               ; send to port 1
CLR RI                  ; get ready to receive next byte
SJMP HERE               ; keep getting data
```

IMPORTANCE OF THE RI FLAG BIT:

It receives the start bit, next bit is the first bit of the character about to be received. When the last bit is received, a byte is formed and placed in SBUF. when stop bit is received, it makes RI = 1 indicating entire character byte has been received and can be read before overwritten by next data. When RI=1, received byte is in the SBUF register, copy SBUF contents to a safe place. After the SBUF contents are copied the RI flag bit must be cleared to 0.

5.3 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.

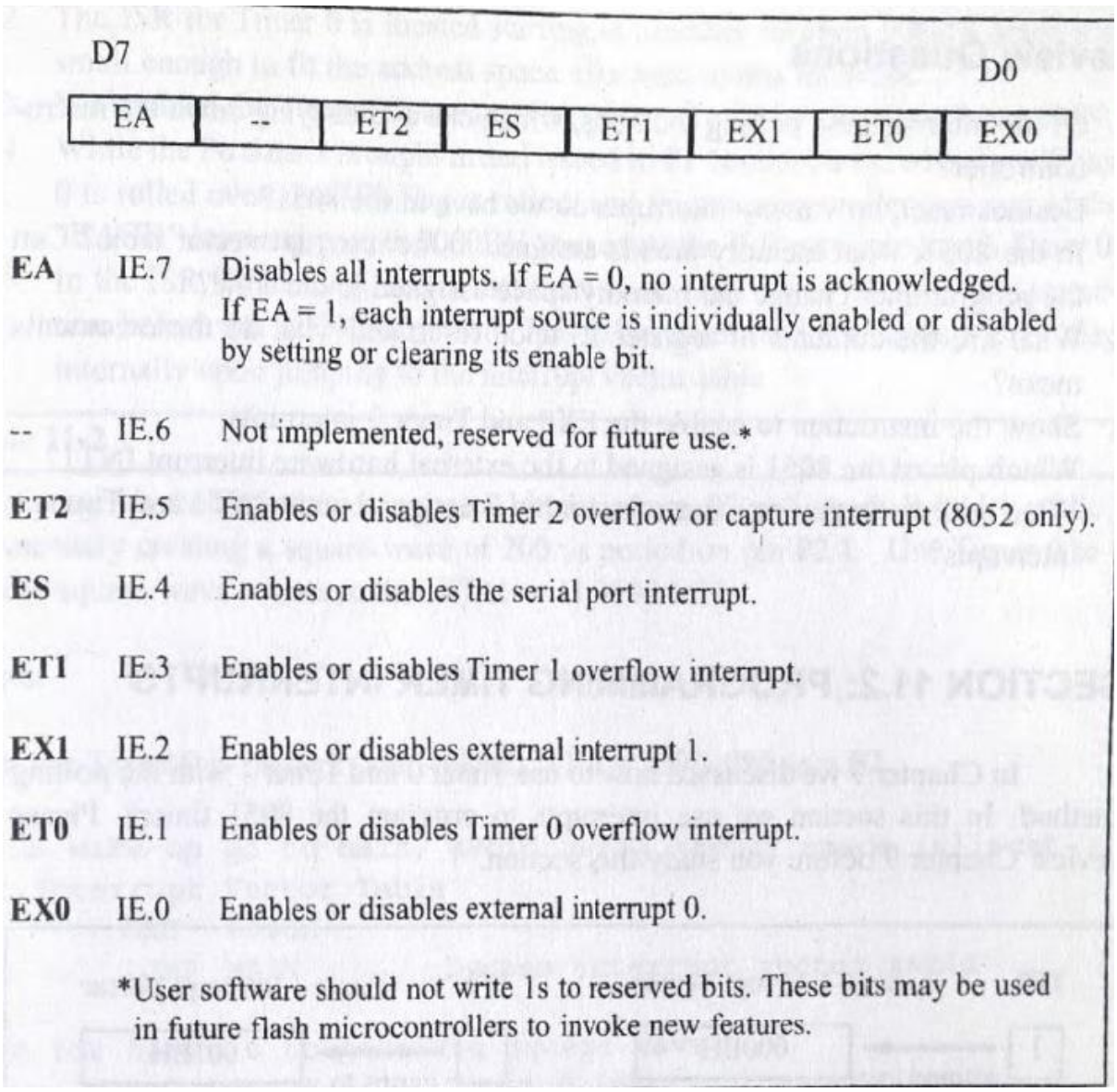
Interrupt	ROM Location (Hex)	Pin	Flag Clearing
Reset	0000	9	Auto
External hardware interrupt 0 (INT0)	0003	P3.2 (12)	Auto
Timer 0 interrupt (TF0)	000B		Auto
External hardware interrupt 1 (INT1)	0013	P3.3 (13)	Auto
Timer 1 interrupt (TF1)	001B		Auto
Serial COM interrupt (RI and TI)	0023		Programmer clears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

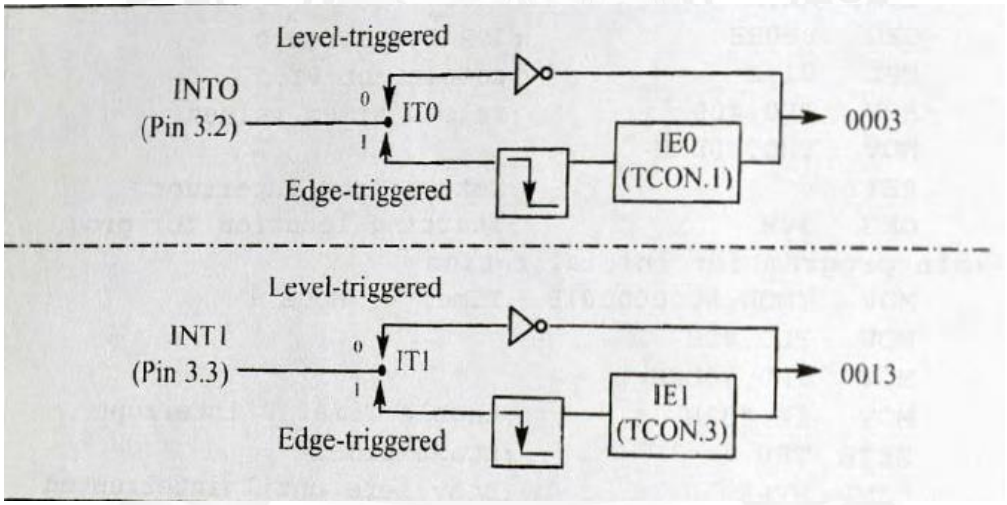


Figure 5.3.2 Activation of INT0 and INT1

[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.

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.

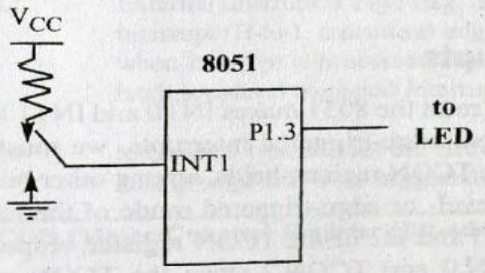


Figure 5.3.3 Example for Level triggered Interrupt

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.327]

SERIAL COMMUNICATION INTERRUPT

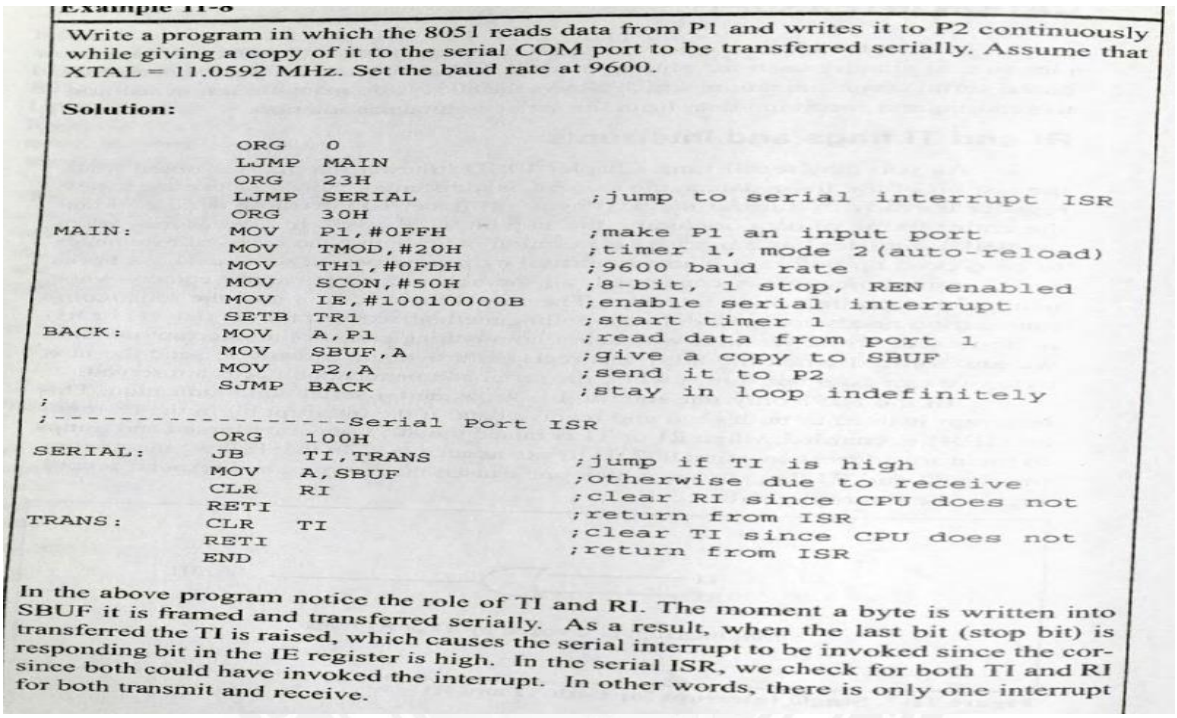
TI (transfer interrupt) is raised when the stop bit is transferred indicating that the SBUF register is ready to transfer the next byte

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 and respond accordingly.



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.

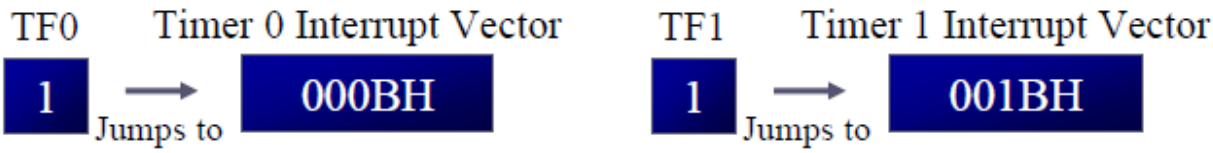
Figure 5.3.4 Example for Serial Communication Interrupt

[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]