## SECTION 11.4: PROGRAMMING THE SERIAL COMMUNICATION INTERRUPT

In Chapter 10 we studied the serial communication of the 8051.

All examples in that chapter used the polling method. In this section we explore interrupt-based serial communication, which allows the 8051 to do many things, in addition to sending and receiving data from the serial communication port.

### RI and TI flags and interrupts

As you may recall from Chapter 10, TI (transfer interrupt) is raised when the last bit of the framed data, the stop bit, is transferred, indicating that the SBUF register is ready to transfer the next byte.
RI (received interrupt), is raised when the entire frame of data, including the stop bit, is received.

In other words, when the SBUF register has a byte, RI is raised to indicate that the received byte needs to be picked up before it is lost (overrun) by new incoming serial data. As far as serial communication is concerned, all the above concepts apply equally when using either polling or an interrupt.

The only difference is in how the serial communication needs are served.

In the polling method, we wait for the flag (TI or RI) to be raised; while we wait we cannot do anything else. In the interrupt method, we are notified when the 8051 has received a byte, or is ready to send the next byte; we can do other things while the serial communication needs are served.

In the 8051 only one interrupt is set aside for serial communication. This interrupt is used to both send and receive 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 address 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.

See Example 11-8.

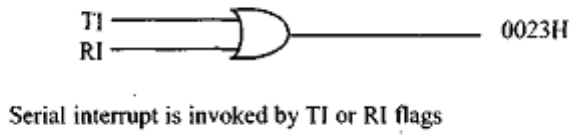


**Figure 11-7. Single Interrupt for Both TI and RI**

### Use of serial COM in the 8051

In the vast majority of applications, the serial interrupt is used mainly for receiving data and is never used for sending data serially. This is like receiving a telephone call, where we need a ring to be notified. If we need to make a phone call there are other ways to remind ourselves and so no need for ringing..

In receiving the phone call, however, we must respond immediately no matter what we are doing or we will miss the call. Similarly, we use the serial interrupt to receive incoming data so that it is not lost.

Look at Example 11-9.

**Example 11-8** Write a program in which the 8051 reads data from PI 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.

## Clearing RI and TI before the RETI instruction

Notice in Example 11-9 that the last instruction before the RETI is the clearing of the RI or TI flags. This is necessary since there is only one interrupt for both receive and transmit, and the 8051 does not know who generated it; therefore, it is the job of the ISR to clear the flag. Contrast this with the external and timer interrupts where it is the job of the 8051 to clear the interrupt flags.

By contrast,

**Example 11-9** Write a program in which the 8051 gets data from PI 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
```

in serial communication the RI (or TI) must be cleared by the programmer using software instructions such as "CLR TI" and "CLR RI" in the ISR. See Example 11-10. Notice that the last two instructions of the ISR are clearing the flag, followed by RETI.

**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 | TI | SCON.1 |
| Timer 2 | TF2 | T2CON.7 (AT89C52) |
| Timer 2 | EXF2 | T2CON.6 (AT89C52) |

Before finishing this section notice the list of all interrupt flags given in Table 11-2. While the TCON register holds four of the interrupt flags, in the 8051 the SCON register has the RI and TI flags.

**Example 11-10** Write a program using interrupts to do the following:

1. Receive data serially and send it to P0,
2. Have port PI read and transmitted serially, and a copy given to P2,
3. Make Timer 0 generate a square wave of 5 kHz frequency on P0.1 (Puerto 0 bit 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
            ORG   100H
SERIAL:     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
```

Referencia FB-POST
https://www.facebook.com/groups/167241170005427/permalink/952800378116165/

http://what-when-how.com/8051-microcontroller/programming-the-serial-communication-interrupt/
** v0626 p04