Serial Port Communication

Instructor

Zhizheng Wu

吴智政

School of Mechatronic Engineering and Automation

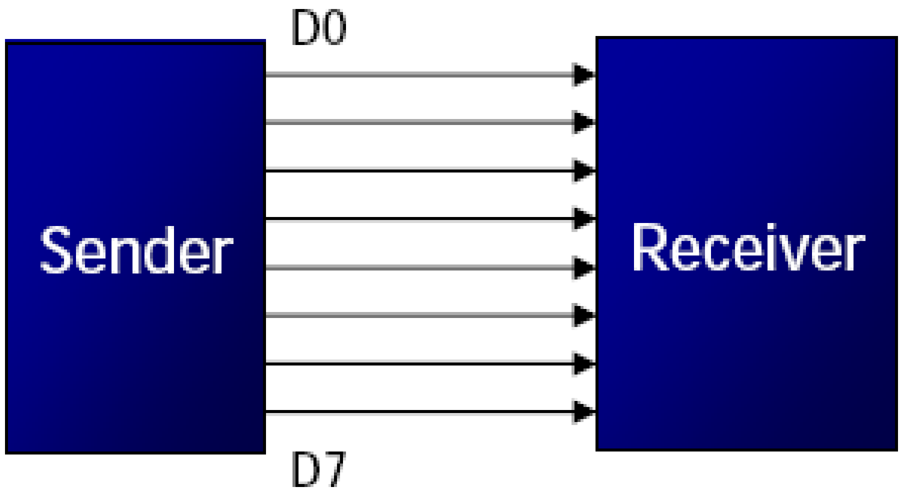
SERIAL PROT COMMUNICATON

- Introduce the serial communication and the RS232 standard.
- Configure the 8051 serial port.
- Communication with the serial port.

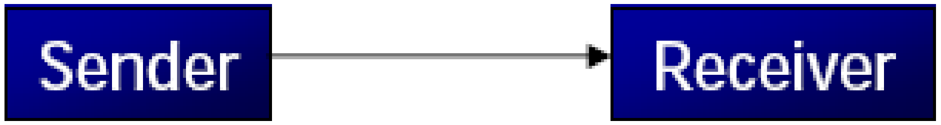
CONCEPTS IN SERIAL COMMUNICATON

- Computers transfer data in two ways:
  - ➢Parallel
    - ☞Often 8 or more lines (wire conductors) are used to transfer data to a device that is only a few feet away
  - ➢Serial
    - ☞To transfer to a device located many meters away, the serial method is used
    - ☞The data is sent one bit at a time

CONCEPTS IN SERIAL COMMUNICATON



Parallel communication



Serial communication

CONCEPTS IN SERIAL COMMUNICATON

- There are several popular types of serial communications. Here are a few worth noting:
  - RS232. Peer-to-peer (i.e. communications between two devices)
  - RS485. Multi-point (i.e. communications between two or more devices)
  - USB (Universal Serial Bus). Replaced RS232 on desktop computers.
  - CAN (Controller Area Network). Multi-point. Popular in the automotive industry.
  - SPI (Serial Peripheral Interface). Developed by Motorola. Synchronous master/slave communications.
  - I2C (Inter-Integrated Circuit). Developed by Philips. Multi-master communications.

CONCEPTS IN SERIAL COMMUNICATON

- Serial data communication uses two methods
  - ➢ Synchronous method transfers a block of data at a time
  - ➢ Asynchronous method transfers a single byte at a time
- It is possible to write software to use either of these methods, but the programs can be tedious and long
  - ➢ There are special IC chips made by many manufacturers for serial communications
    - ✓UART (Universal Asynchronous Receiver Transmitter)
    - ✓USART (Universal Synchronous Asynchronous Receiver Transmitter)

CONCEPTS IN SERIAL COMMUNICATON

- UART (pronounced “You Art”) is an industry acronym that stands for Universal Asynchronous Receiver Transmitter. It is the interface circuitry between the microprocessor and the serial port. This circuitry is built in to the 8051 microcontroller.

- The UART is responsible for breaking apart bytes of data and transmitting it one bit at a time (i.e. serially). Likewise, the UART receives serialized bits and converts them back into bytes. In practice, it’s a little more complicated, but that’s the basic idea.

CONCEPTS IN SERIAL COMMUNICATON

- The UART, however, doesn’t operate at the line voltages required by the RS232 standard. The UART operates at TTL voltage levels (i.e. 0 to 5V). For noise immunity and transmission length, the RS232 standard dictates the transmission of bits at a higher voltage range and different polarities (i.e. typically -9V to +9V). An external transceiver chip such as MAX232 is needed.

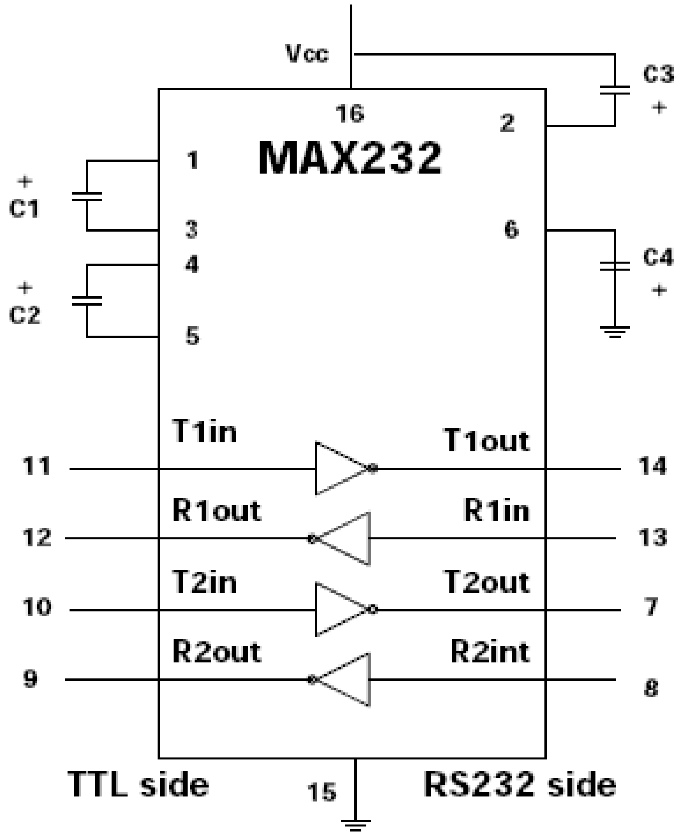
☞ Binary 0: UART: 0V RS232: 3-25V

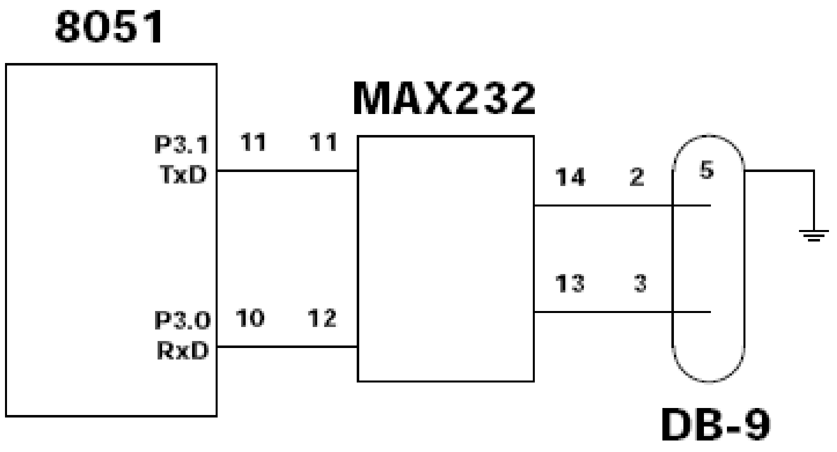
☞ Binary 1: UART: 5V RS232: -3V to -25V

CONCEPTS IN SERIAL COMMUNICATON

- A line driver such as the MAX232 chip is required to convert UART TTL levels to RS232 voltage levels, and vice versa
- 8051 has two pins that are used specifically for transferring and receiving data serially
  - These two pins are called TxD and RxD and are part of the port 3 group (P3.0 and P3.1)
  - These pins are TTL compatible; therefore, they require a line driver to make them RS232 compatible

CONCEPTS IN SERIAL COMMUNICATON

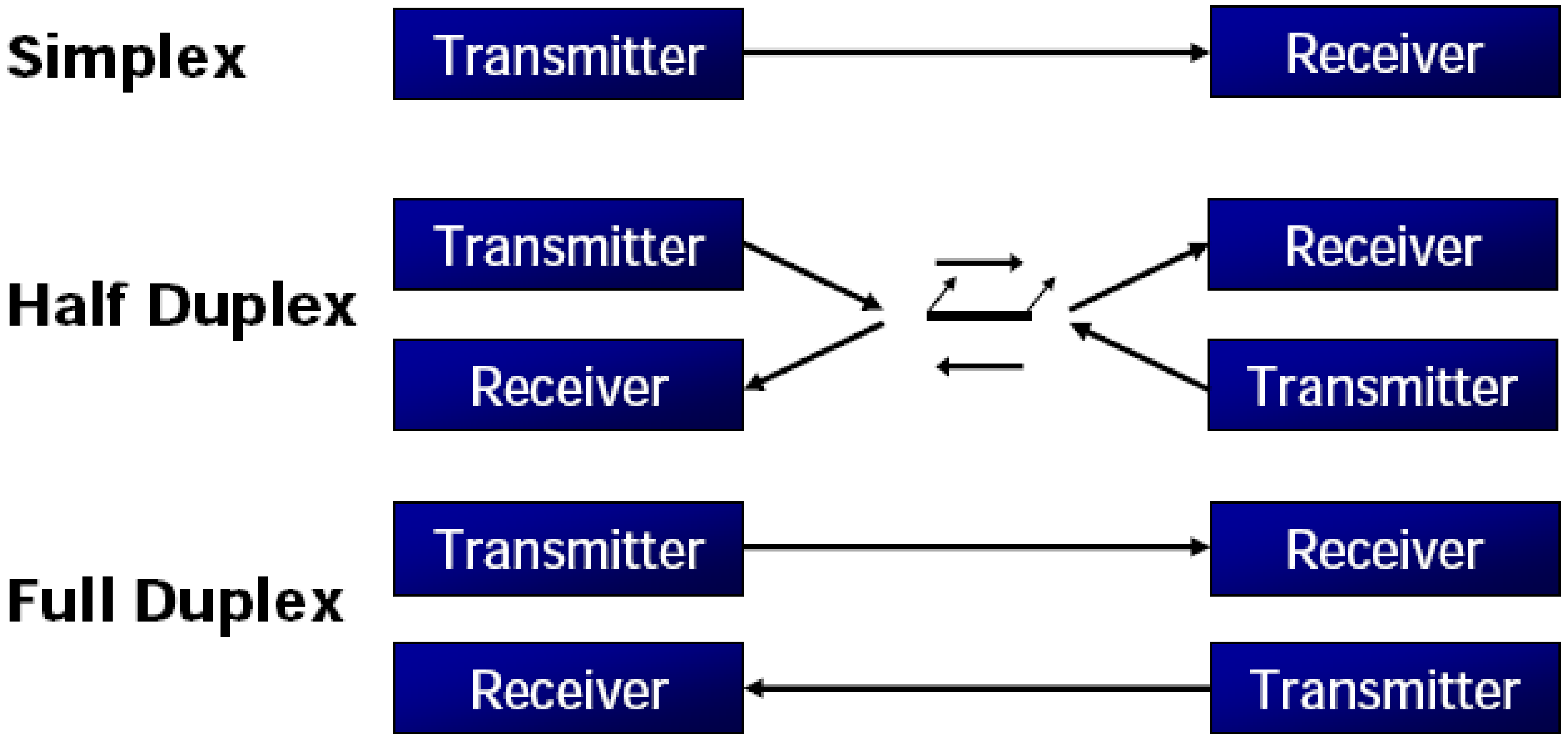




CONCEPTS IN SERIAL COMMUNICATON

- Three ways for data transmitted and received
  - ➢Simplex transmission
    - ☞data only can be transmitted in one direction
  - ➢Half duplex transmission
    - ☞data is transmitted one direction a time
  - ➢Full duplex transmission
    - ☞data can go both ways at a time

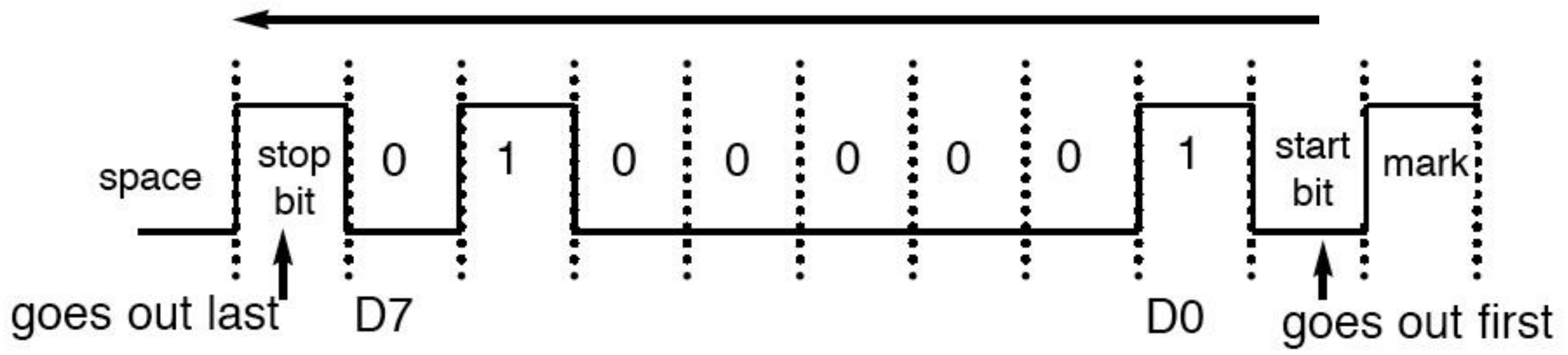
CONCEPTS IN SERIAL COMMUNICATON



CONCEPTS IN SERIAL COMMUNICATON

- A protocol is a set of rules agreed by both the sender and receiver on
  - How the data is packed
  - How many bits constitute a character
  - When the data begins and ends
  - Each character is placed in between start and stop bits, this is called framing
  - Start bit: 0, Stop bit: 1
  - When there is no transfer, the signal is 1 (high) → Mark.

CONCEPTS IN SERIAL COMMUNICATON



Data sequence in the serial communication

☻ In some systems in order to maintain data integrity, the parity bit of the character byte is included in the data frame

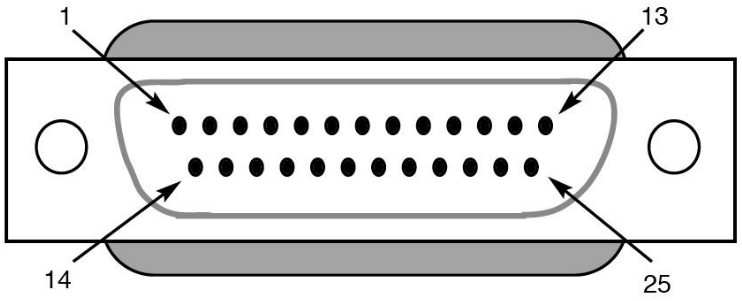
CONCEPTS IN SERIAL COMMUNICATON

- The rate of data transfer in serial data communication is stated in bps (bits per second)
- Another widely used terminology for bps is baud rate
  - As the modem terminology it is defined as the number of signal changes per second
  - In modems, there are occasions when a single change of signal transfers several bits of data
- In asynchronous data communication, the baud rate is usually limited to 100K bps.

CONCEPTS IN SERIAL COMMUNICATON

- The hardware interfaces for serial communication
  - ➢RS232 connector DB-25
    - ☞An interfacing standard RS232 was set by the Electronics Industries Association (EIA) in 1960
  - ➢RS232 connector DB-9
    - ☞ Since not all pins are used in PC cables, IBM introduced the DB-9 version of the serial I/O standard

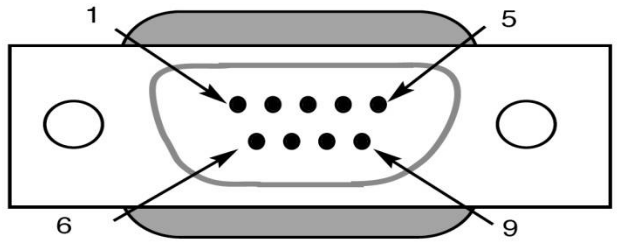
CONCEPTS IN SERIAL COMMUNICATON



RS232 DB-25

Pin	Description	Pin	Description
1	Protective ground	14	Secondary transmitted data
2	Transmitted data (TxD)	15	Transmitted signal element timing
3	Received data (RxD)	16	Secondary receive data
4	Request to send (-RTS)	17	Receive signal element timing
5	Clear to send (-CTS)	18	Unassigned
6	Data set ready (-DSR)	19	Secondary receive data
7	Signal ground (GND)	20	Data terminal ready (-DTR)
8	Data carrier detect (-DCD)	21	Signal quality detector
9/10	Reserved for data testing	22	Ring indicator (RI)
11	Unassigned	23	Data signal rate select
12	Secondary data carrier detect	24	Transmit signal element timing
13	Secondary clear to send	25	Unassigned

CONCEPTS IN SERIAL COMMUNICATON



RS232 DB-9

Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

CONCEPTS IN SERIAL COMMUNICATON

- DTR (data terminal ready)

  - When terminal is turned on, it sends out signal DTR to indicate that it is ready for communication

- DSR (data set ready)

  - When DCE (data communication equipment) is turned on and has gone through the self-test, it assert DSR to indicate that it is ready to communicate

- RTS (request to send)

  - When the DTE device has byte to transmit, it assert RTS to signal the modem that it has a byte of data to transmit

- CTS (clear to send)

  - When the modem has room for storing the data it is to receive, it sends out signal CTS to DTE to indicate that it can receive the data now

CONCEPTS IN SERIAL COMMUNICATON

- DCD (data carrier detect)

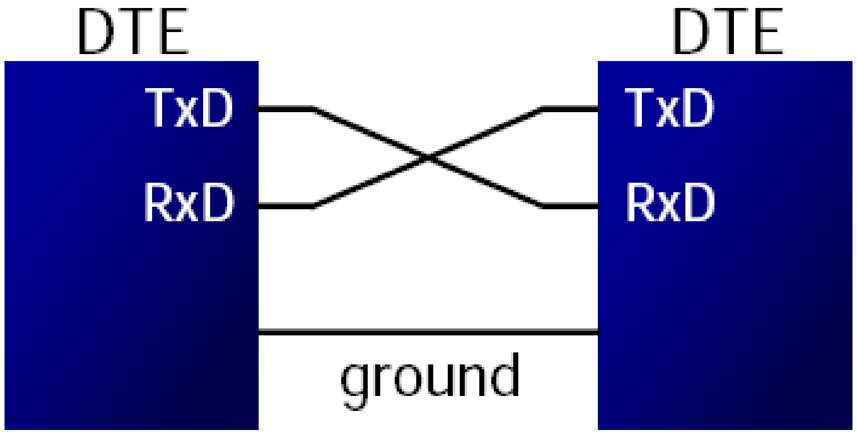
  - The modem asserts signal DCD to inform the DTE that a valid carrier has been detected and that contact between it and the other modem is established

- RI (ring indicator)

  - An output from the modem and an input to a PC indicates that the telephone is ringing. It goes on and off in synchronous with the ringing sound

CONCEPTS IN SERIAL COMMUNICATON

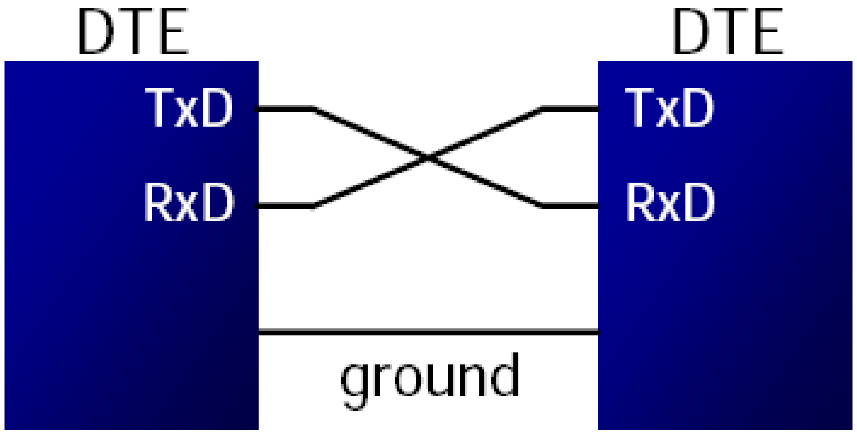
• The simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground



☻ DTE (data terminal equipment) refers to terminal and computers that send and receive data

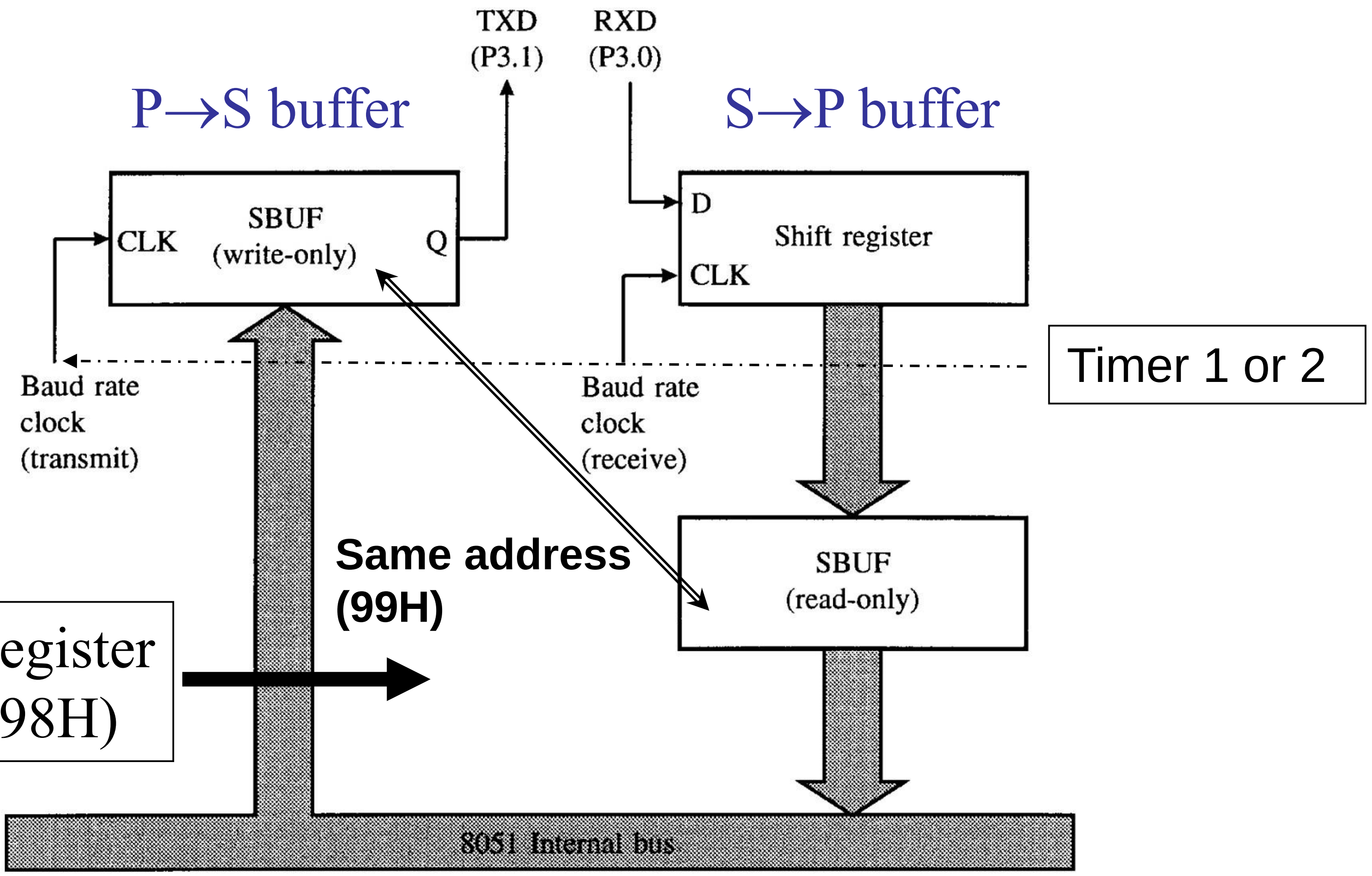
CONFIGURE THE SERIAL PORT

• The simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground



☻ DTE (data terminal equipment) refers to terminal and computers that send and receive data

CONFIGURE THE SERIAL PORT



Control register
SCON(98H)

Serial port block diagram

CONFIGURE THE SERIAL PORT

- SBUF is an 8-bit register used solely for serial communication
  - ➢ For a byte data to be transferred via the TxD line, it must be placed in the SBUF register
  - ➢ The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD line
- SBUF holds the byte of data when it is received by 8051 RxD line
  - ➢ When the bits are received serially via RxD, the 8051 deframes it by eliminating the stop and start bits, making a byte out of the data received, and then placing it in SBUF

CONFIGURE THE SERIAL PORT

- Focus on communication between PC and 8051. Need to make sure baud rate for the two devices match
- Baud rate in 8051
  - ➢The baud rate is programmable
  - ➢8051 divides the crystal frequency by 12 to get the machine cycle frequency
  - ➢For XTAL=11.0592 MHz, machine cycle frequency = 921.6 KHz
  - ➢UART divides the machine cycle frequency by 32, and uses the resulting signal in Timer 1 to set the baud rate (28,800Hz – Default value upon reset)
  - ➢To use timer 1 to set the baud rate in mode 2 (8-bit mode – auto-reload)

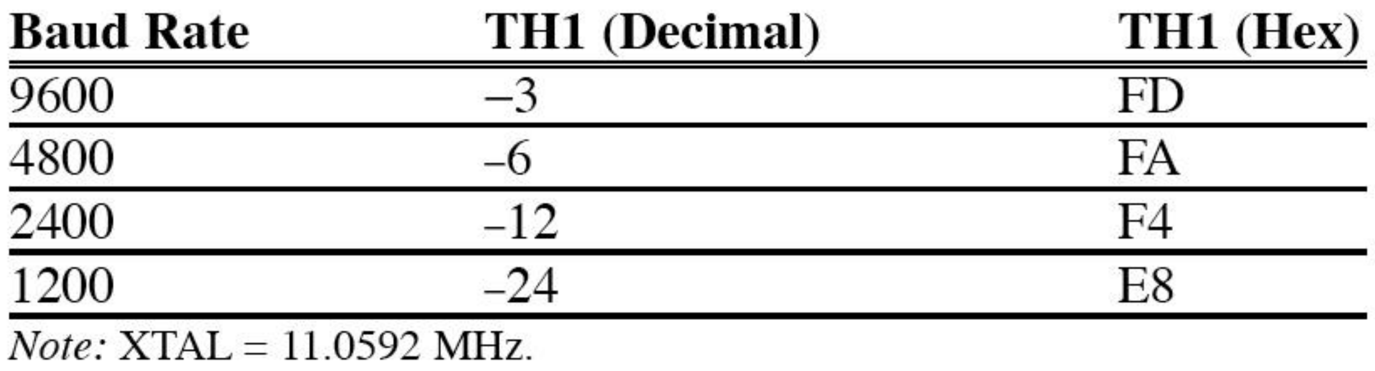
COMMUNICATION WITH THE SERIAL PORT

- There are two ways to increase the baud rate of data transfer
  - ➢To use a higher frequency crystal
  - ➢To change a bit in the PCON register
- PCON register is an 8-bit register (Address: 87H)
  - ➢When 8051 is powered up, SMOD is zero
  - ➢We can set it to high by software and thereby double the baud rate

SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**

CONFIGURE THE SERIAL PORT

- Load the TH1 with one of the numbers in the table below:



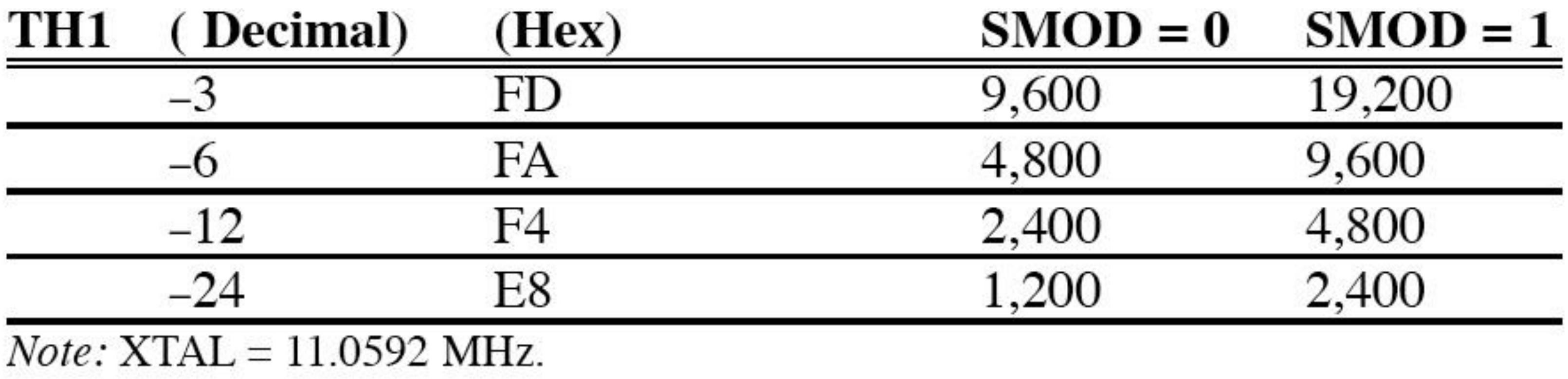
Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	E8

Note: XTAL = 11.0592 MHz.

$$BR = 2^{SMOD} \times \frac{f_{osc}}{12 \times 32} \div \left(2^8 - TH1\right)$$

$$TH1 = 2^8 - 2^0 \times \frac{11.0592 \times 10^6}{384 \times 1200} = 256 - 24 = E8H$$

COMMUNICATION WITH THE SERIAL PORT



TH1 (Decimal)	(Hex)	SMOD = 0	SMOD = 1
–3	FD	9,600	19,200
–6	FA	4,800	9,600
–12	F4	2,400	4,800
–24	E8	1,200	2,400

Note: XTAL = 11.0592 MHz.

Example:

```
MOV A, PCON          ; place a copy of PCON in ACC
SETB ACC.7           ; make D7=1
MOV PCON, A          ; now SMOD=1 without changing any
                     other bits
```

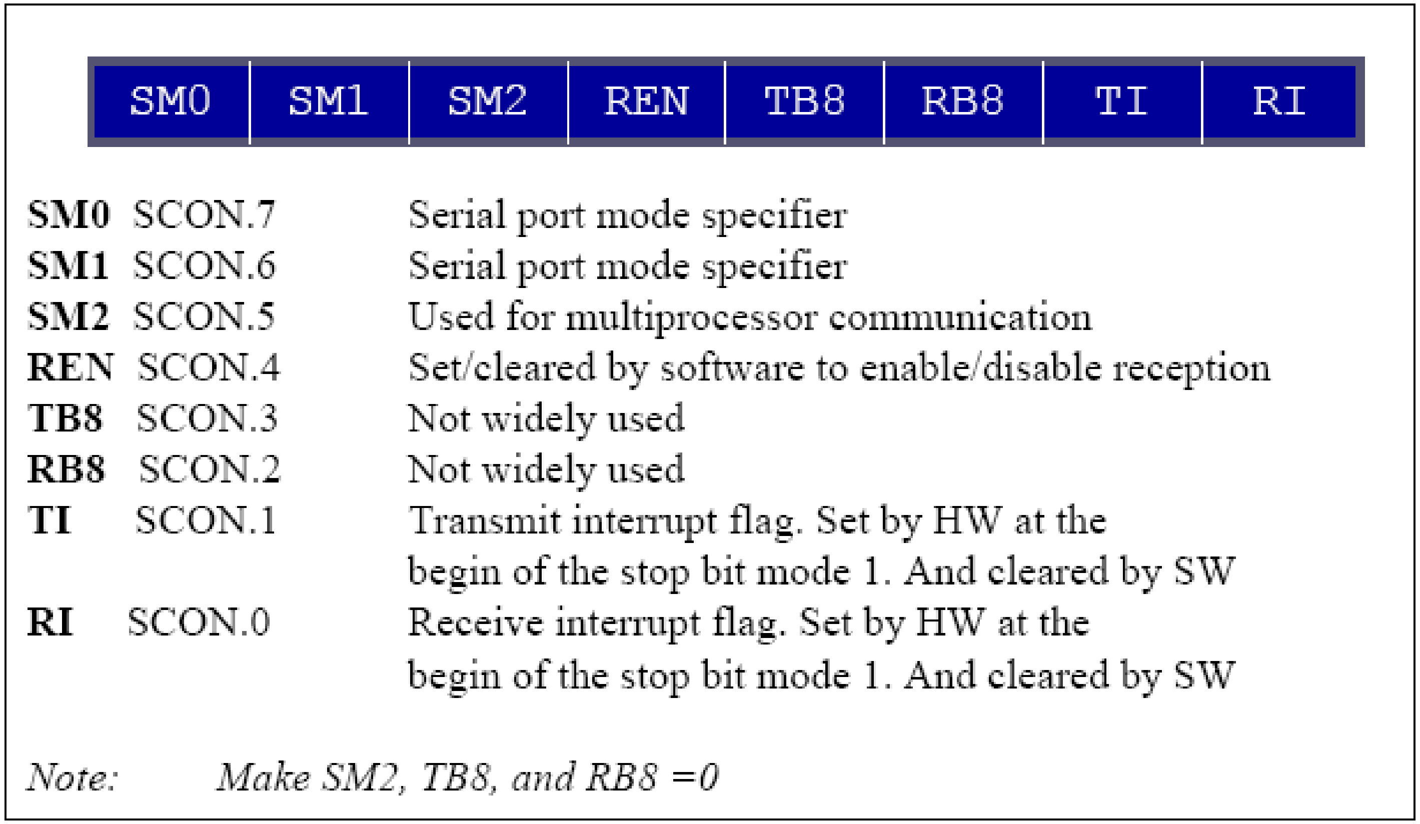
CONFIGURE THE SERIAL PORT

Example:

```
MOV SBUF, # 'D'          ;load SBUF=44h, ASCII for 'D'
MOV SBUF, A              ;copy accumulator into SBUF
MOV A, SBUF              ;copy SBUF into accumulator
```

CONFIGURE THE SERIAL PORT

- SCON (Serial Control) Register
  - ➢SCON is an 8-bit special function register (Address: 98H) used to program the start bit, stop bit, and data bits for data framing



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
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 HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

Note: Make SM2, TB8, and RB8 =0

CONFIGURE THE SERIAL PORT

- SM0, SM1: Determine the framing data by specifying the number of bits/character, and the start and stop bits

SM0	SM1	
0	0	Serial mode 0, fixed baud rate
0	1	Serial mode 1, 10-bit data (8-bit data, 1 stop bit, 1 start bit), adjustable baud rate
1	0	Serial mode 2, 11-bit data, fixed baud rate
1	1	Serial mode 3, 11-bit data, adjustable baud rate

CONFIGURE THE SERIAL PORT

- Only mode 1 will be tested
- SM2: Enables multiprocessing capability of the 8051
- REN: Receive enable bit, SCON.4 bit. When high (**SETB SCON.4**), it allows the 8051 to receive data on the RxD pin. When 0 (**CLR SCON.4**), the receiver is disabled
- TB8: Used for serial modes 2 and 3
- RB8: gets a copy of the stop bit when an 8-bit data is received
- TI: Transmit interrupt bit. TI is set when the 8051 finishes the transfer of the 8-bit data, indicating the microcontroller is ready to transfer another byte. TI bit is raised at the beginning of the stop bit
- RI: Receive interrupt. RI is set to indicate that a byte has been received. RI is set halfway through the stop bit

COMMUNICATION WITH THE SERIAL PORT

- Programming the 8051 to transfer data serially
  1. The TMOD register is loaded with the value 20H (Timer 1 in mode 2: 8-bit auto-reload)
  2. TH1 is loaded with one of the values needed to set the baud rate for serial data transfer (assuming XTAL = 11.0592MHz)
  3. SCON is loaded with the value 50H, indicating serial mode 1 (in this mode, 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 the **CLR TI** instruction
  6. Write the character byte to be transferred serially into SBUF register
  7. The TI flag bit is monitored with the use of the instructions "**JNB TI, xx**" to see if the character has been transferred completely
  8. Go to step 5 to transfer the next Character

COMMUNICATION WITH THE SERIAL PORT

Example: Write a program for the 8051 to transfer letter “A” serially at 4800 baud, continuously.

```
        MOV TMOD, #20H       ; Timer 1, mode 2 (auto-reload)
        MOV  TH1, #-6        ; 4800 bps baud rate
        MOV  SCON, #50H      ; 8-bit, 1 stop, REN enabled
        SETB TR1             ; Start Timer 1
AGAIN:MOV SBUF, #”A”         ; Letter “A” to be transferred
HERE:  JNB  TI, HERE         ; wait for the last bit
        CLR  TI              ; clear TI for next char
        SJMP AGAIN           ; keep sending A
```

COMMUNICATION WITH THE SERIAL PORT

Example: Write a program for the 8051 to transfer “YES” serially at 9600 baud, 8-bit data, 1 stop bit, do this 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 A, #”Y”                 ;transfer “Y”
    ACALL TRANS
    MOV A, #”E”                    ;transfer “E”
    ACALL TRANS
    MOV A, #”S”                    ;transfer “S”
    ACALL TRANS
    SJMP AGAIN                     ;keep doing it
    ;serial data transfer subroutine
TRANS: MOV SBUF, A                 ;load SBUF
HERE: JNB TI, HERE      ;wait for the last bit
    CLR TI                         ;get ready for next byte
    RET
```

COMMUNICATION WITH THE SERIAL PORT

- Programming the 8051 to receive data serially

1. The TMOD register is loaded with the value 20H (Timer 1 in mode 2: 8-bit auto-reload).
2. TH1 is loaded with one of the values needed to set the baud rate for serial data transfer (assuming XTAL = 11.0592MHz)
3. SCON is loaded with the value 50H, indicating serial mode 1 (in this mode, 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 the "**CLR RI**" instruction.
6. The RI flag bit is monitored with the use of the instructions "**JNB RI, xx**" to see if the entire character has been received.
7. When RI is raised, SBUF has the byte. Its contents are moved into a different memory location.
8. Go to step 5 to receive the next Character.

COMMUNICATION WITH THE SERIAL PORT

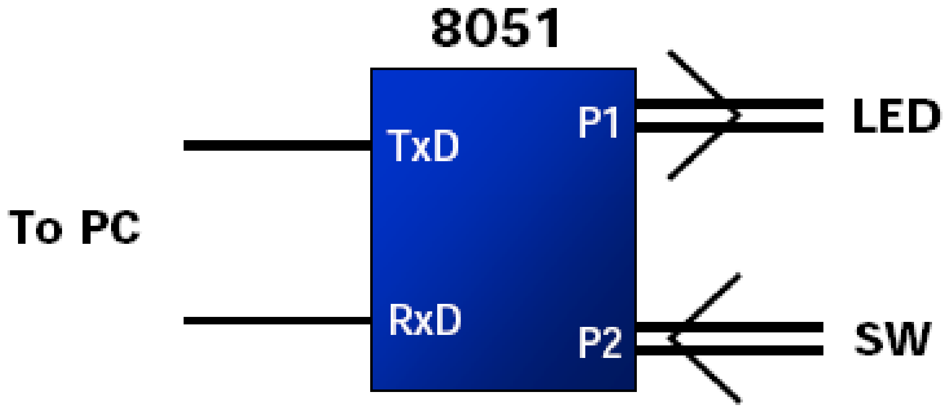
Example: Write a program for the 8051 to receive bytes of data serially, and put them in P1, set the baud rate at 4800, 8-bit data, and 1 stop bit

```
        MOV TMOD, #20H          ; Timer 1, mode 2 (auto-reload)
        MOV  TH1, #-6           ; 4800 bps 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             ; save incoming byte in A
        MOV P1, A               ; send to port 1
        CLR  RI                 ; get ready to receive next byte
        SJMP HERE               ; keep getting data
```

COMMUNICATION WITH THE SERIAL PORT

Example:

Assume that the 8051 serial port is connected to the PC. P1 and P2 of the 8051 are connected to LEDs and switches, respectively. Write an 8051 program to (a) send to PC the message "We Are Ready", (b) receive any data send by PC and put it on LEDs connected to P1, and (c) get data on switches connected to P2 and send it to PC serially. The program should perform part (a) once, but parts (b) and (c) continuously, use 4800 baud rate.



COMMUNICATION WITH THE SERIAL PORT

Solution:

```
    ORG 0
    MOV P2, #0FFH       ;make P2 an input port
    MOV TMOD, #20H      ;timer 1, mode 2
    MOV TH1, #0FAH      ;4800 baud rate
    MOV SCON, #50H      ;8-bit, 1 stop, REN enabled
    SETB TR1            ;start timer 1
    MOV DPTR, #MYDATA ;load pointer for message
H_1:    CLR A
    MOV A, @A+DPTR      ;get the character
    JZ B_1                          ;if last character get out
    ACALL SEND                      ;otherwise call transfer
    INC DPTR            ;next one
    SJMP H_1            ;stay in loop
B_1:    MOV A, P2                   ;read data on P2
    ACALL SEND                      ;transfer it serially
```

………..continuing………………….

COMMUNICATION WITH THE SERIAL PORT

```
        ACALL RECV                      ;get the serial data
        MOV P1, A                 ;display it on LEDs
        SJMP B_1                  ;stay in loop indefinitely
        ;----serial data transfer. ACC has the data------
SEND:   MOV SBUF, A                     ;load the data
H_2:    JNB TI, H_2                     ;stay here until last bit gone
        CLR TI                          ;get ready for next char
        RET                             ;return to caller
        ;----Receive data serially in ACC----------------
RECV: JNB RI,  RECV                     ;wait here for char
        MOV A, SBUF                     ;save it in ACC
        CLR RI                          ;get ready for next char
        RET                             ;return to caller
        ;-----The message---------------
MYDATA: DB “We Are Ready”, 0
        END
```