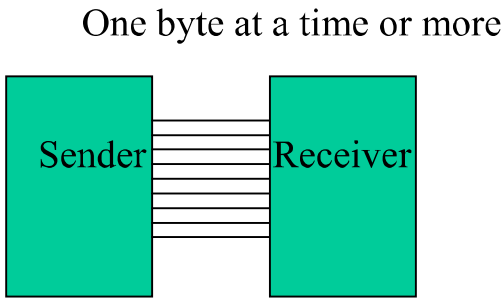
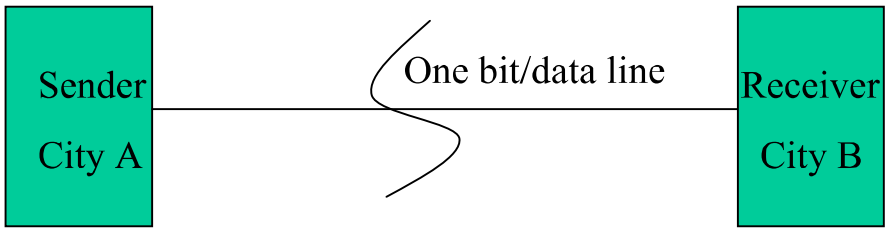
8051 Serial Communications

Parallel vs. Serial

- Parallel Communication (Printer)
  - Fast, but distance cannot be great.
  - Expensive

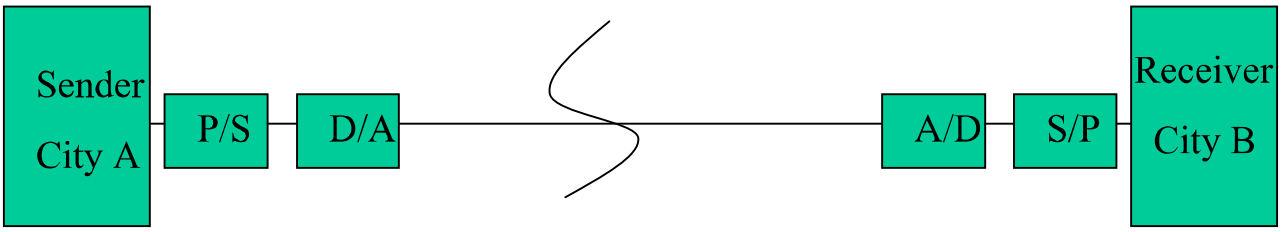


- Serial Communication (Telephone line)
  - Long distance
  - cheaper



Serial Communications

- Serial Communication (Telephone line)
- P/S: Parallel-in-Serial-out Shift Register
- S/P: Serial-in-Parallel-out Shift Register
- D/A: Digital-Analog Converter
- A/D: Analog-Digital Converter

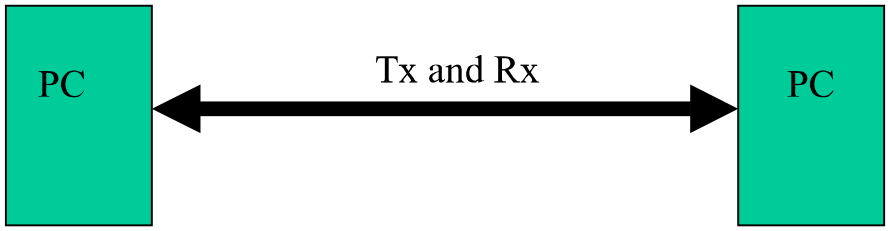


Asynchronous vs. Synchronous

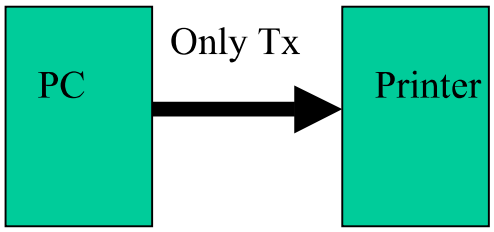
- Two methods of Serial Communication
  - Synchronous: Transfer block of data (characters) at a time.
  - Asynchronous: Transfer ***a single byte*** at a time.
- You could write S/W to use either of these methods, but the programs can be tedious and long. So, H/W is developed instead.
- This H/W is called UART or USART
- UART: Universal Asynchronous Receiver-Transmitter
- USART: Universal Synchronous-Asynchronous Receiver-Transmitter
- The 8051 chip has a built-in UART.

Duplex vs. Simplex

- Duplex

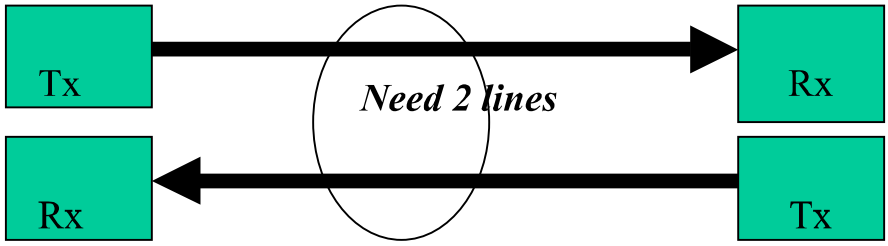


- Simplex (only transmit)

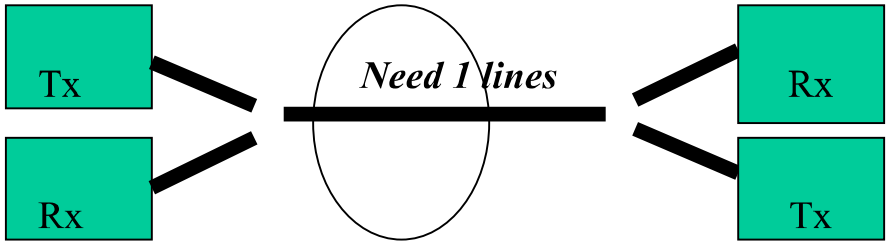


Duplex

- Full Duplex: Data is transmitted both way at the same time.

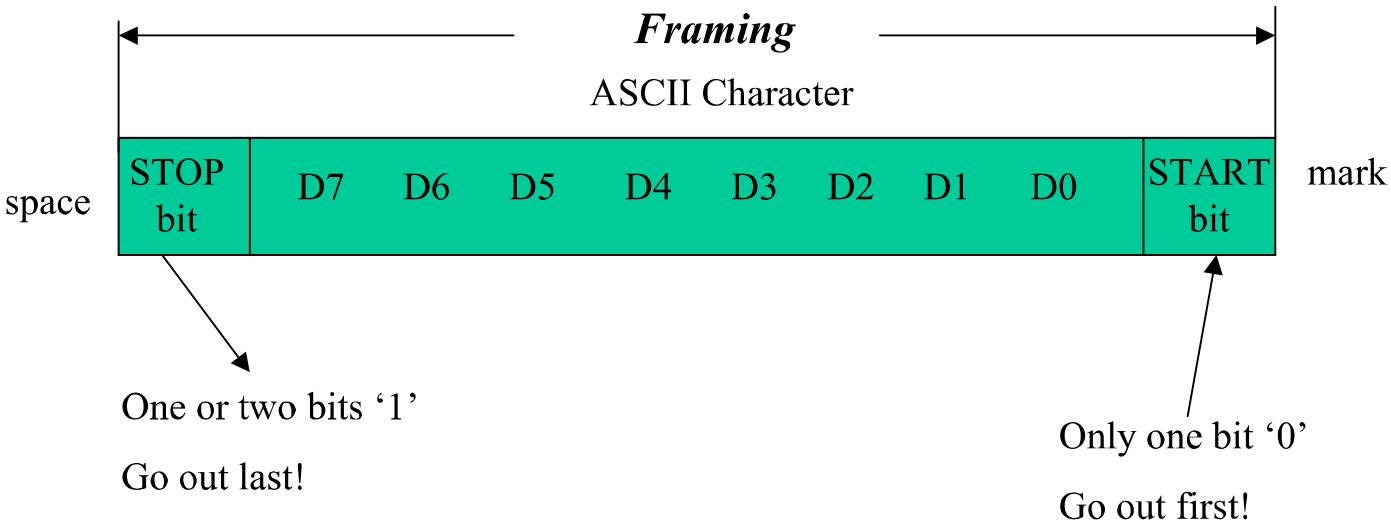


- Half Duplex: Data is transmitted one way at a time.



Asynchronous Communications

- Asynchronous serial communication is widely used for character-oriented (1 byte) transmissions.
  - Synchronous serial communication is widely used for block-oriented (multiple bytes) transmissions.
- Start and stop bits

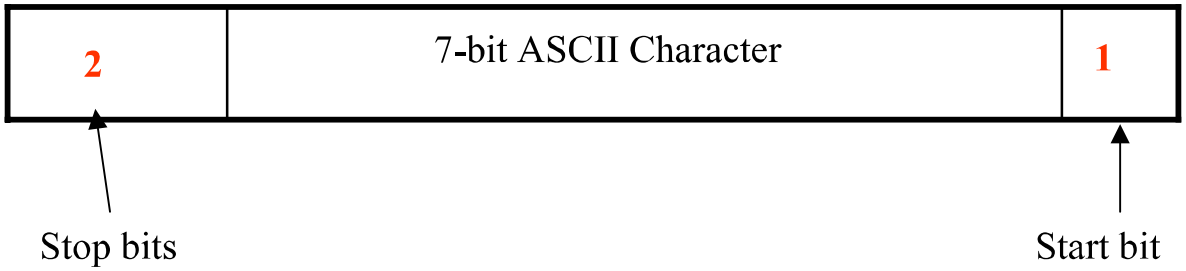


Asynchronous

- New System: ASCII character A (0100 0001B) → 8-bit
  - When there is no transfer, mark is '1' and space is '0'.

Space =0	1	0	1	0	0	0	0	0	1	0	Mark =1

- Older System: ASCII is 7-bit
  - Due to the slowness of receiving mechanical devices, 2 stop bits were used to give the device sufficient time to organize itself.



Parity Bit

- Even-Parity bit system
  - 01001011 ➔ # of '1'=4 ➔ P=0
- Odd-Parity bit system
  - 01001011 ➔ # of '1'=4 ➔ P=1
- UART chips allow programming of the parity bit for odd-, even-, and no-parity options.

Data Transfer Rate

- Data Transfer Rate
  - Bits per second (bps) is also called Baud rate (the number of signals changes per second)
  - IBM PC/XT ➔ 100 ~ 9600 bps
  - Pentium PC ➔ 56k bps (recent MODEM)
- Asynchronous data communication, the baud rate is limited to 100,000 bps.
- Interfacing standard: RS232

RS232

- PC: COM1 (Mouse), COM2: (MODEM & RS232)
- RS232 is set by Electronics Industries Association (EIA) in 1960.
- 1963: RS232A
- 1965: RS232B
- 1969: RS232C
- RS232 is a serial I/O interfacing standard; however, since the standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible.
- TTL is Transistor-Transistor Logic family (Bipolar transistor) in 1968

Line Driver: MAX232

- RS232 is not compatible to TTL. For this reason, to connect any RS232 to a microcontroller system, we must use “Voltage Converters” such as MAX232 to convert the TTL logic levels to the RS232 voltage levels, and vice versa.
- MAX232 IC chips is referred to as “Line Driver”.

RS232 Connector: DB-25

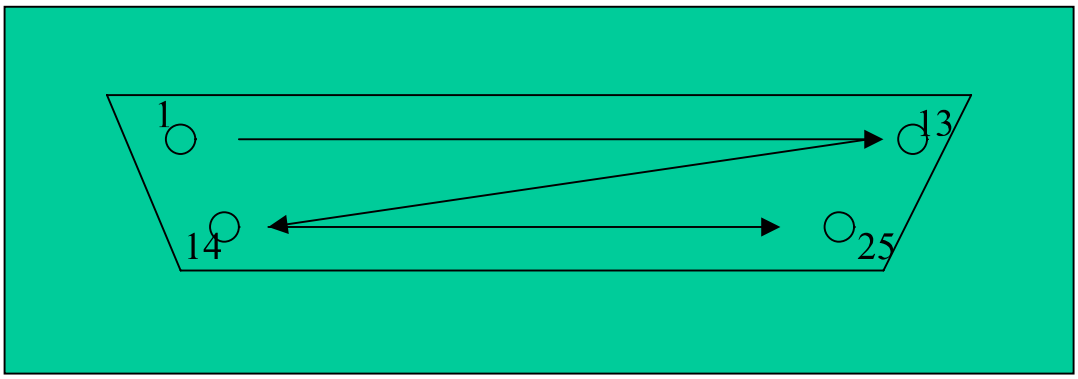
- DB-25 Connector
  - DB-25P (Plug)
  - DB-25S (Socket)

Pin 1: Ground

Pin 2: TxD

Pin 3: RxD

:

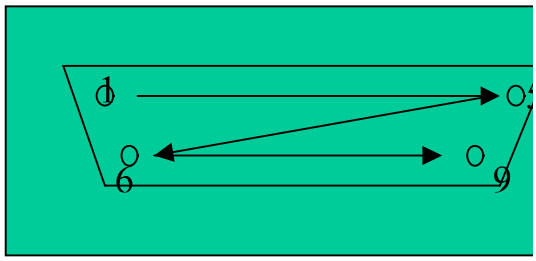


- DTE: Data Terminal Equipment: RS232
- DCE: Data Communication Equipment: MODEM

RS232 Connector: DB-9

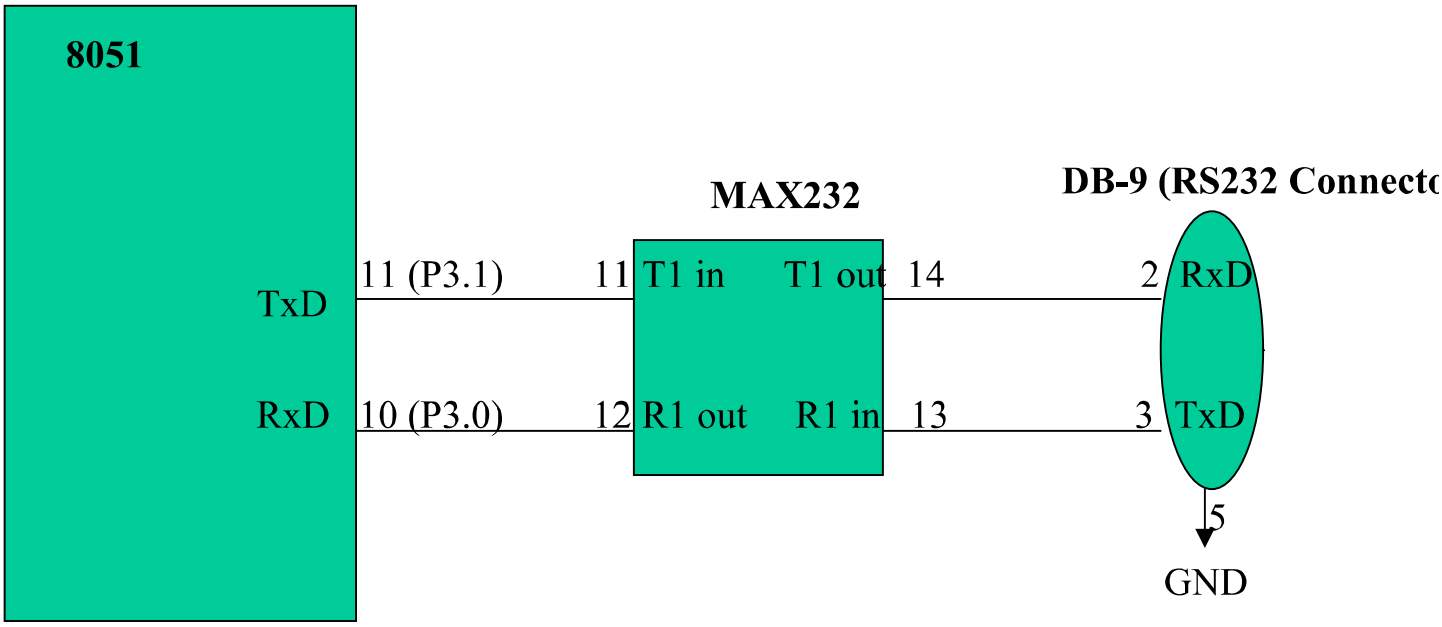
- DB-9 Connector

Pin	function	description
1	!DCD	Data Carrier Detect
2	RxD	Receive
3	TxD	Transmit
4	DTR	Data Terminal Ready
5	GND	Ground
6	!DSR	Data Set Ready
7	!RTS	Request To Send
8	!CTS	Clear To Send
9	RI	Ring Indicator



8051 Connection to RS232

- Since RS232 standard is not TTL compatible, so we need MAX232 (line driver)

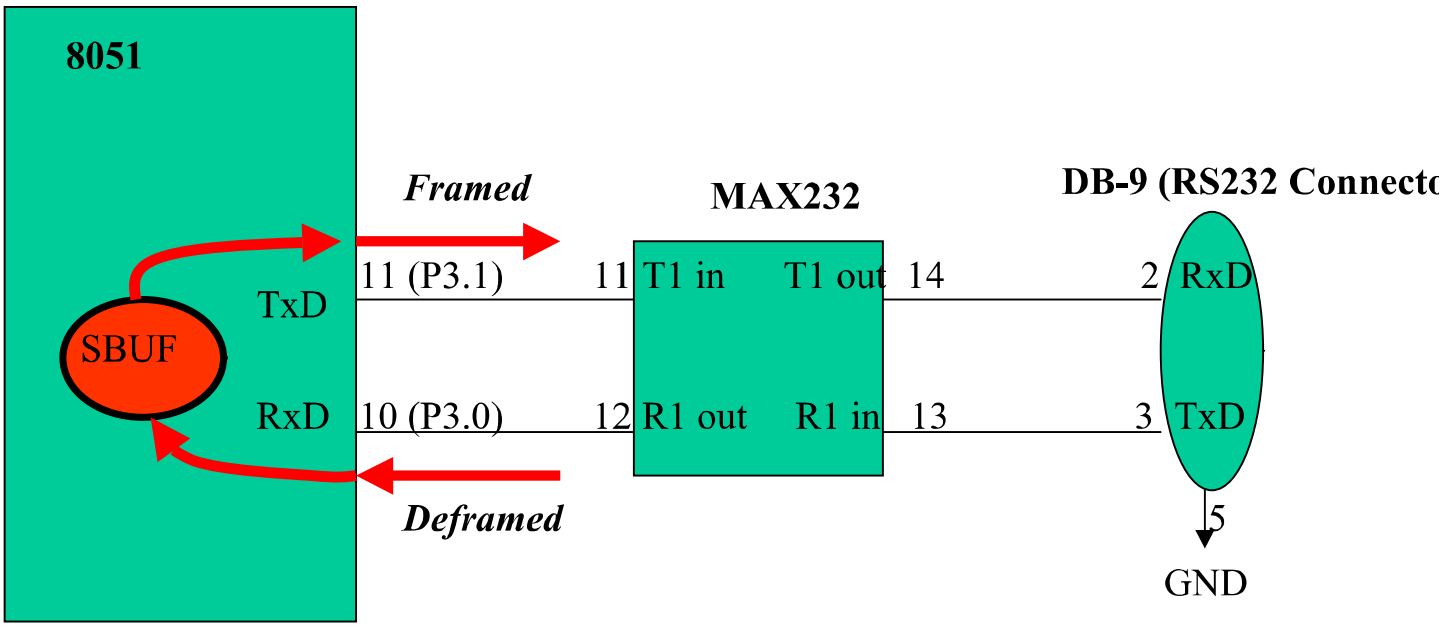


Baud Rate

- Q1: With XTAL=11.0592MHz, find the TH1 value needed to have the following baud rate: (a) 9600, (b) 2400, and (c) 1200.

SBUF Register

- SBUF Register: For a byte of data to be transferred via the TxD line, it must be placed in the SBUF.
- SBUF holds the byte of data when it is received by the 8051's RxD line.



SCON Register

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

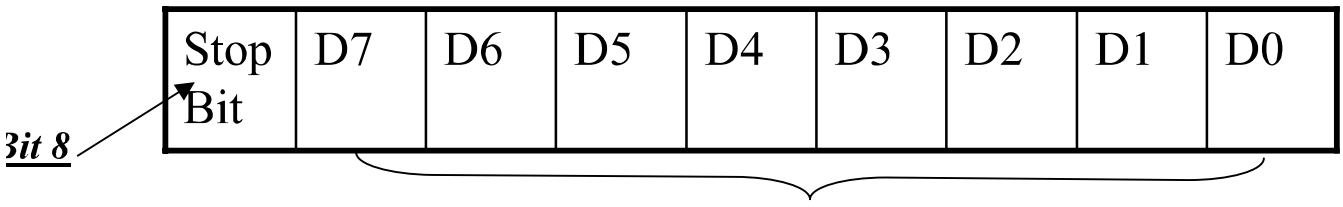
SCON

- Mode 0, 2, and 3 are ready used today.
- Mode 1 is compatible with the COM port of IBM PC.
- Mode 1 allows the baud rate to be variable and is set by Timer 1 of 8051.

SM0	SM1	Mode	
0	0	0	
0	***1***	***1***	***8 bit data, 1 stop bit , 1 start bit***
1	0	2	
1	1	3	

SCON Register

- SM2 ➔ enable the multiprocessing capability of the 8051. Here we make SM2=0.
- REN: Receive Enable
  - REN=1: it allows the 8051 to receive data on the RxD pin. If you want the 8051 to both transfer and receive data, REN must be set to '1'. ***SETB SCON.4***
  - REN=0: make the receiver is disabled.
- TB8: Transfer bit 8
  - Is used for serial modes 2 and 3, so here we make TB8=0.



SCON Register

- RB8: Receive bit 8
  - Is used for serial modes 2 and 3, so here we make RB8=0.
- TI: Transmit Interrupt (this is a flag bit)
  - When 8051 finish the transfer of the 8-bit character, it raises the TI flag to indicate that it is ready to transfer another byte.
- RI: Receive Interrupt
  - When 8051 receive data via RxD, it get rid of the start and stop bits (deframed procedure) and places the byte in the SBUF.
  - Then it raises the RI flag to indicate that a byte has been received and should be picked up before it is lost.

Programming (Transfer)

1. MOV TMOD, #20H ; Timer 1, mode 2
2. TH1 is loaded to set the baud rate.
3. MOV SCON, #50H

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

SCON

REN=1: allows TxD and RxD

4. SETB TR1 ; Run Timer 1
5. MOV SBUF, #'D'
6. Loop: JNB TI, Loop ; Monitor TI
7. CLR TI
8. Repeat step 5 for next character.

Examples

- Q2: Write a program for the 8051 to transfer letter ‘A’ serially at 4800 baud rate, continuously.
- Q3: Write a program to transfer the message “Yes” serially at 9600 baud, do this continuously.

Programming (Receive)

1. MOV TMOD, #20H ; Timer 1, mode 2
2. TH1 is loaded to set the baud rate.
3. MOV SCON, #50H

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

SCON

REN=1: allows TxD and RxD

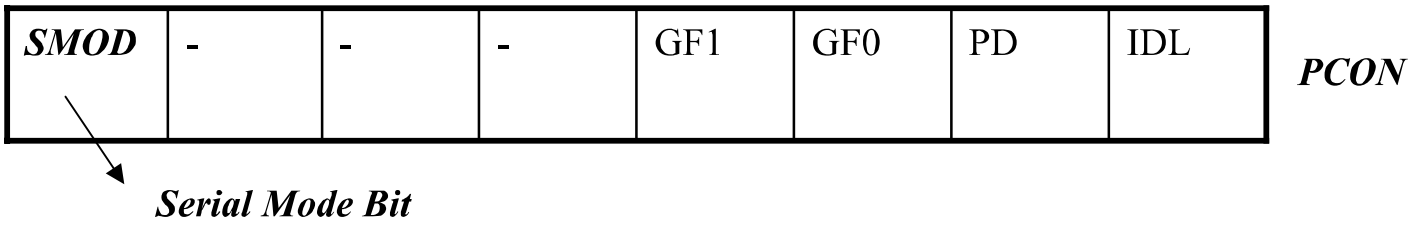
4. SETB TR1 ; Run Timer 1
5. Loop: JNB RI, Loop ; Monitor RI
6. MOV A, SBUF
7. CLR RI
8. Repeat step 5 for next character.

Examples

- Q4: Program the 8051 to receive bytes of data serially, and put them in P1. Set the baud rate at 4800.
- Q5: Write a program to (a) send to the PC the message "We Are Ready", (b) receive any data sent by the PC and put it ob LEDs connected to P1, and (c) get data on switches and sent it to PC. The program should perform part (a) once, but parts (b) and (c) continuously. Use the 4800 baud rate.

Doubling the Baud Rate

- Method 1: Use a high XTAL
  - This method is infeasible since XTAL is fixed.
  - If you change XTAL, and then it is not compatible to PC.
- Method 2: Change a bit in the PCON register



- When 8051 is powered up, D7 of PCON =0, we could set D7 of PCON '1' to double the baud rate.
- Note that PCON is not bit addressible, so we need to do the following

```
MOV A, PCON
SETB ACC.7
MOV PCON, A
```

Baud Rate for SMOD

- Baud rate for SMOD = ‘0’

  11.0592MHz/12 = 921.6KHz

  921.6KHz/32 = 28800Hz

- Baud rate for SMOD = ‘1’

  11.0592MHz/12 = 921.6KHz

  921.6KHz/16 = 57600Hz

Baud Rate for SMOD

Baud Rate for SMOD=0	Baud Rate for SMOD=1 (Doubling the Baud Rate)	TH1 (decimal)	TH1 (Hex)
9600	19200	-3	FD
4800	9600	-6	FA
2400	4800	-12	F4
1200	2400	-24	E8

Examples

- Q6: Assume XTAL=11.0592MHz, for the following program, state (a) what this program does (b) compute the frequency used by timer 1 to set the baud rate, and (c) find the baud rate of the data transfer.
- Q7: Find the TH1 value (in both decimal and hex) to set the baud rate to each of the following (a) 9600, and (b) 4800 if SMOD='1'. Assume XTAL=11.0592MHz.
- Q8: Find the baud rate if TH1 =-2, SMOD =1, XTAL=11.0592MHz. Is this baud rate supported by IBM PC