Unit 3- Part2

8051 Serial Communication Interface

Outline

- Serial Port Operation
- Setting the Serial Port Mode
- Setting the Baud Rate
- Writing to the Serial Port
- Reading from the Serial Port
- Programming example in assembly
- Programming examples in Embedded ‘C’

Basics of Serial Communication

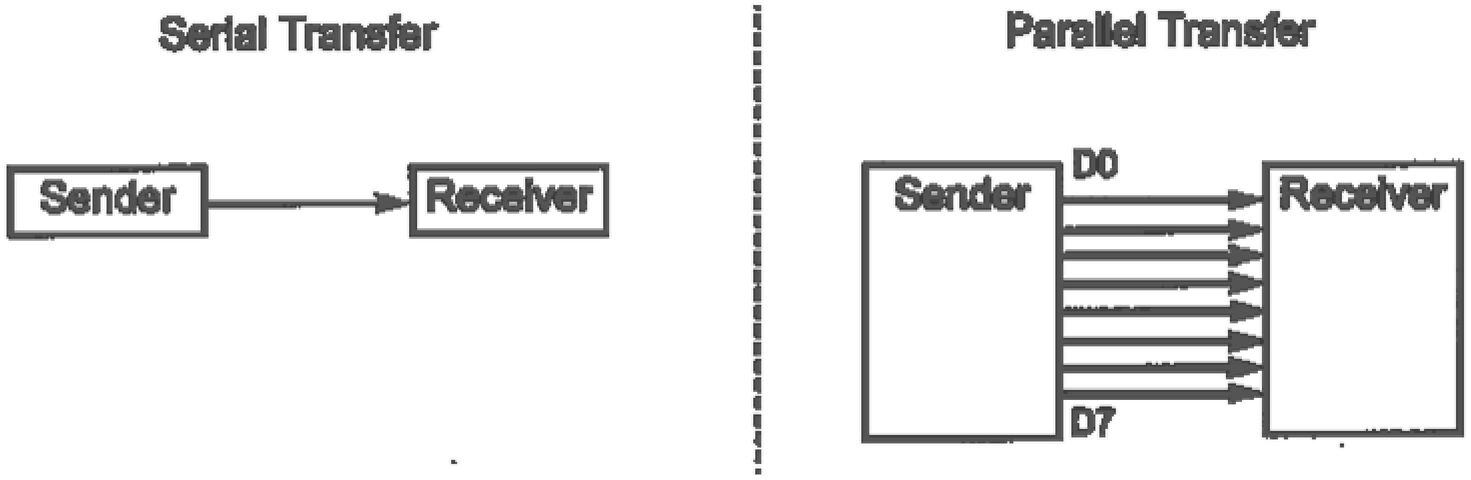
Parallel: expensive - short distance – fast – no modulation

Serial :cheaper– long (two different cities by modem)-slow

Basics of Serial Communication

- When a microprocessor communicates with the outside world, it provides the data in byte-sized chunks.
- In some cases, such as printers, the information is simply grabbed from the 8-bit data bus and presented to the 8-bit data bus of the printer.
- This can work only if the cable is not too long, since long cables diminish and even distort signals.
- Further, an 8-bit data path is expensive

- For these reasons, serial communication is used for transferring data between two systems located at distances of hundreds of feet to millions of miles apart.



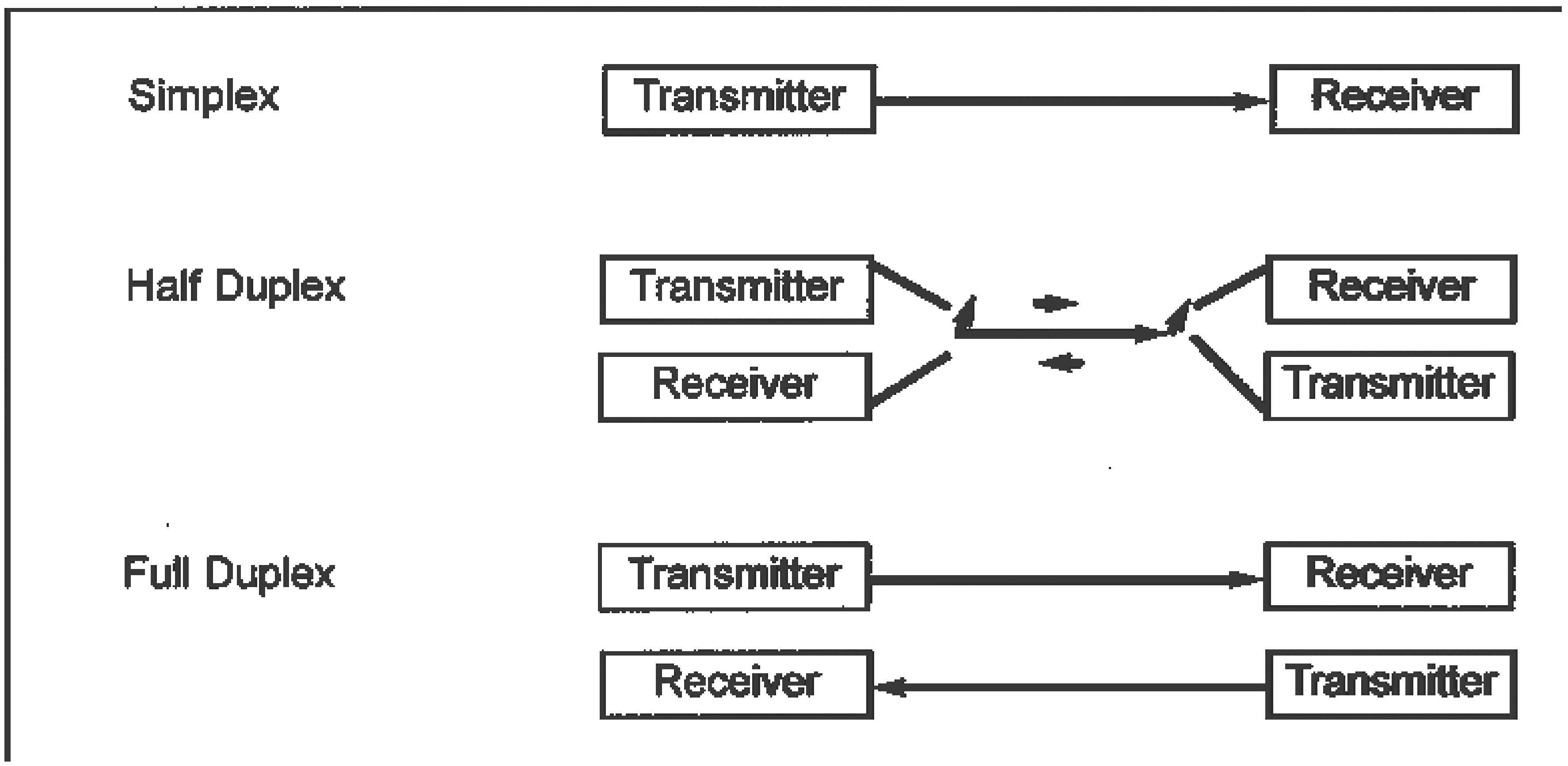
- Since serial communication uses single data line instead of the 8-bit data line of parallel communication, it is much cheaper and also it enables two computers located in two different cities to communicate over the telephone
- For serial communication to work, byte of data must be converted to serial bits using a parallel in serial out shift register, then it can be transmitted over a single data line.
- At the receiving end, there must be a serial in parallel out shift register to receive the serial data and pack them into a byte

- If the data is to be transferred on the telephone line, it must be converted from 1’s and 0’s to audio tones, which are sinusoidal shaped signals
- This conversion is performed by a peripheral device called a modem, (modulator/demodulator)
- When the distance is short, the digital signal can be transferred as it is on a simple wire and requires no modulation
- Ex:data transfer from keyboard to the motherboard

- For long distance data transfers using communication lines such as telephone, serial data communication requires a modem to modulate (convert 0’s and 1’s to audio tones) and demodulate(converting from audio tones to 0s and 1s)

- Serial data communication uses 2 methods, synchronous and asynchronous
- The synchronous method transfers a block of data(characters) at a time, while the 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
- Hence special ICs , UART and USART are used for serial data communications

Half duplex and full duplex transmission



- In data transmission if the data can be transmitted and received, it is duplex communication
- In simplex transmissions such as with printers, the computer only sends data
- Duplex transmission can be half or full duplex, depending on whether or not the data transfer can be simultaneous
- If data is transmitted one way at a time, it is referred to as half duplex
- If the data can go both ways at the same time, it is full duplex

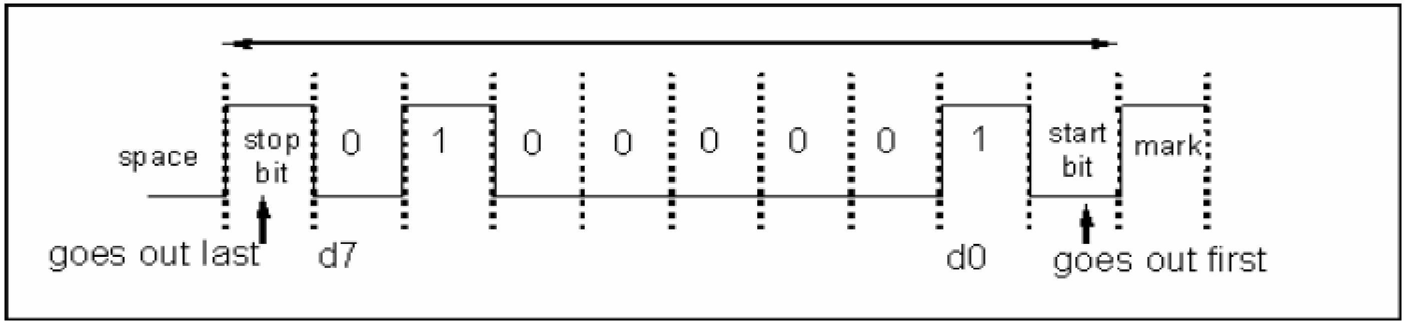
Asynchronous serial communication and data framing

- The data coming in at the receiving end of the data line in a serial data transfer is all 0s and 1s
- It is difficult to make sense of the data unless sender and receiver agree on a set of rules, a protocol, on how the data is packed, how many bits constitute a character, and when the data begins and ends

Start and stop bits

- Asynchronous serial data communication is widely used for character-oriented transmissions, while block-oriented data transfers use the synchronous
- In the asynchronous method, each character is placed between start and stop bits
- This is called framing
- Data is packed between a start and a stop bit
- Start bit is always one bit while the stop can be one or two bits
- Start bit is always low(0) while stop bit is always high(1)

Start and stop bits



Framing ASCII 'A' (41H)

- When there is no transfer, the signal is 1 – referred to as mark. The 0 is referred to as space
- The transmission begins with a start bit followed by d0, which is the LSB, then the rest of the bits until d7, and finally the one stop bit is sent indicating the end of the character "A"

Data transfer rate

- The rate of data transfer in serial communication is stated in bps. (bits per second)
- Another widely used terminology for bps is baud rate(number of signal changes per second)
- In modems, single change of signal, sometimes transfers several bits of data
- As far as the conductor wire is concerned, the baud rate and bps are the same

RS 232 standards

- To allow compatibility among data communication equipment made by various manufacturers, an interfacing standard called RS232 was set by the Electronics Industries Association(EIA) in 1960
- Most widely used interfacing standard
- Used in PCs and numerous types of equipments

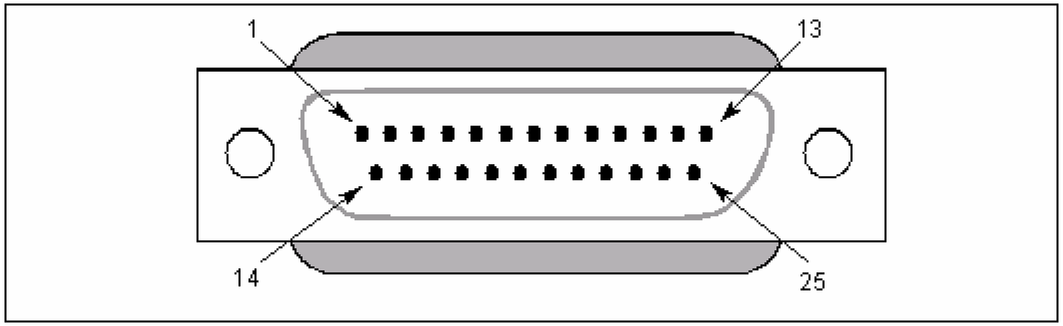
- Standard for serial comm (COM port)
- Logic 1 : represented by -3 to -25 volt
- Logic 0 : represented by +3 to +25 volt
- Hence to connect any RS232 to a microcontroller system, we must use voltage converters such as MAX232 to convert the TTL logic levels to the RS232 voltage levels and vice versa
- The baud rate of the 8051 must matched the baud rate of the pc
- PC standard baud rate
  - 2400-4800-9600-14400-19200-28800-33600-57600

- Connectors

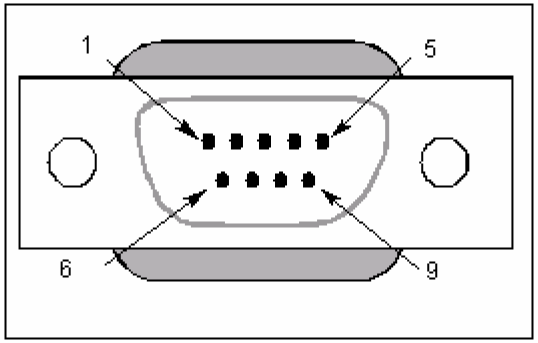
– Minimally, 3 wires: RxD, TxD, GND

– Could have 9-pin or 25-pin

RS232 Connector

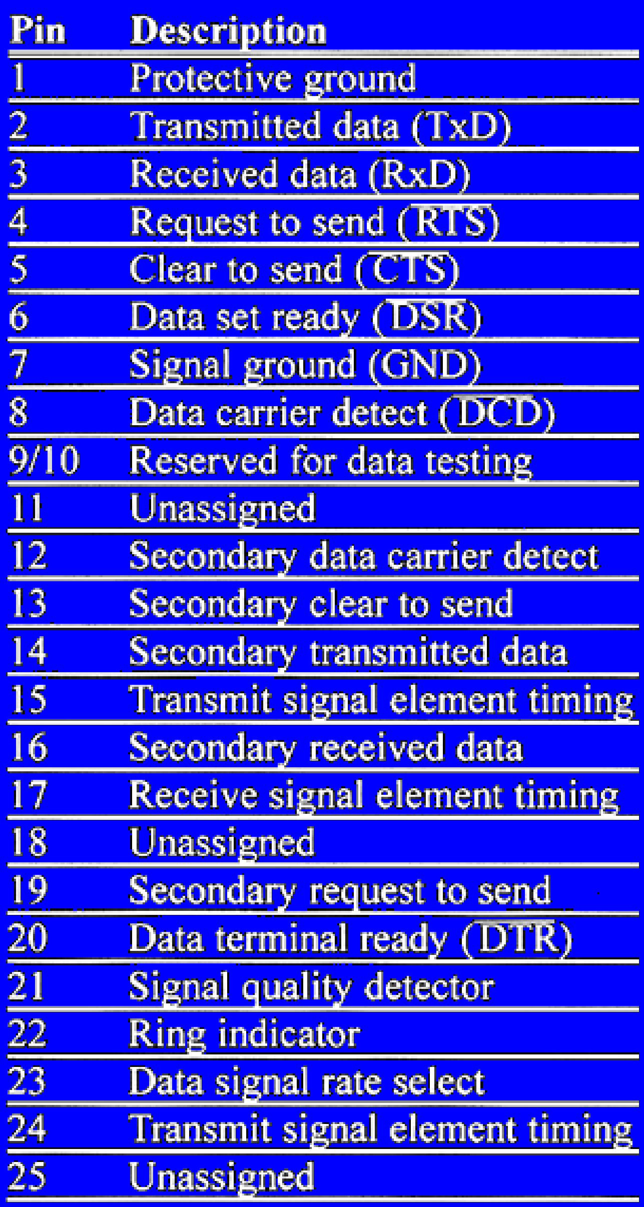


DB-25, 25 pin connector



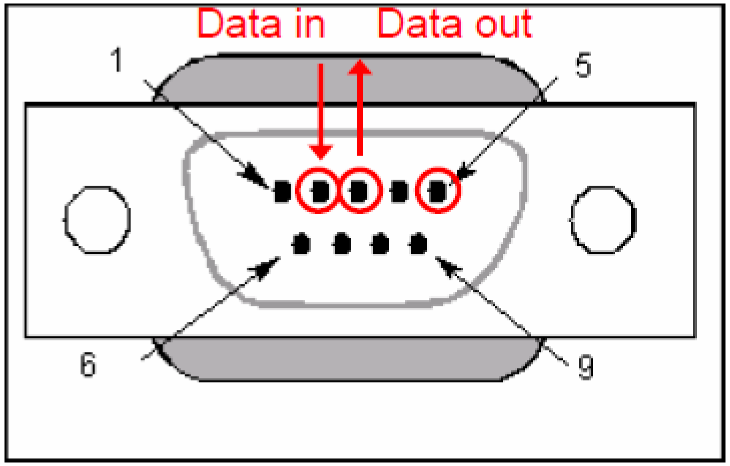
DB-9, 9 pin connector

Table 10-1: RS232 Pins (DB-25)



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

RS232 Connector *(cont'd)*



DB-9

9-Pin Connector

IBM PC DB-9 Signals

Pin 1 – Data Carrier Detect (DCD)
Pin 2 – Received Data (RxD)
Pin 3 – Transmitted Data (TxD)
Pin 4 – Data Terminal Ready (DTR)
Pin 5 – Signal Ground (GND)
Pin 6 – Data Set Ready (/DSR)
Pin 7 – Request to Send (/RTS)
Pin 8 – Clear to Send (/CTS)
Pin 9 – Ring Indicator (RI)

Data communication classification

- Current terminology classifies data communication equipment as DTE or DCE (Data terminal/communication Equipment)
- DTE refers to the terminals and computers that send and receive data
- DCE refers to communication equipment, such as modems, that are responsible for transferring the data
- The simplest connection between a PC and a microcontroller requires a minimum of 3 pins, TXD, RXD and ground

- DTR-data terminal ready-DTE sends out DTR signal to indicate that it is ready for communication
- DSR-data set ready-DCE(modem) asserts DSR to indicate that it is ready to communicate
- RTS-request to send-When the DTE has a byte to transmit, asserts RTS to signal the modem that it has a byte of data to transmit
- CTS-clear to send-DCE sends out this signal to DTE, in response to RTS, when it has room for storing the data
- DCD-Carrier data detect-the modem asserts this signal to inform DTE (PC) that a valid carrier has been detected and that contact b/w it and the modem is established
- RI-ring indicator-o/p from the modem(DCE) and an i/p to a PC(DTE) indicates that the telephone is ringing

8051 connection to RS232

- 8051 has 2 pins for transferring and receiving data serially
- TxD pin 11 of the 8051 (P3.1)
- RxD pin 10 of the 8051 (P3.0)
- These pins are TTL compatible
- Hence require a line driver to make them RS232 compatible
- One such driver is the MAX232 chip

MAX232

- Since the RS232 is not compatible with today’s microprocessors and microcontrollers, we need a line driver (voltage converter) to convert the RS232’s signals to TTL voltage levels that will be acceptable to the 8051’s TxD and RxD pins
- The MAX232 converts from RS232 voltage levels to TTL voltage levels, and vice –versa

MAX232 or MAX233

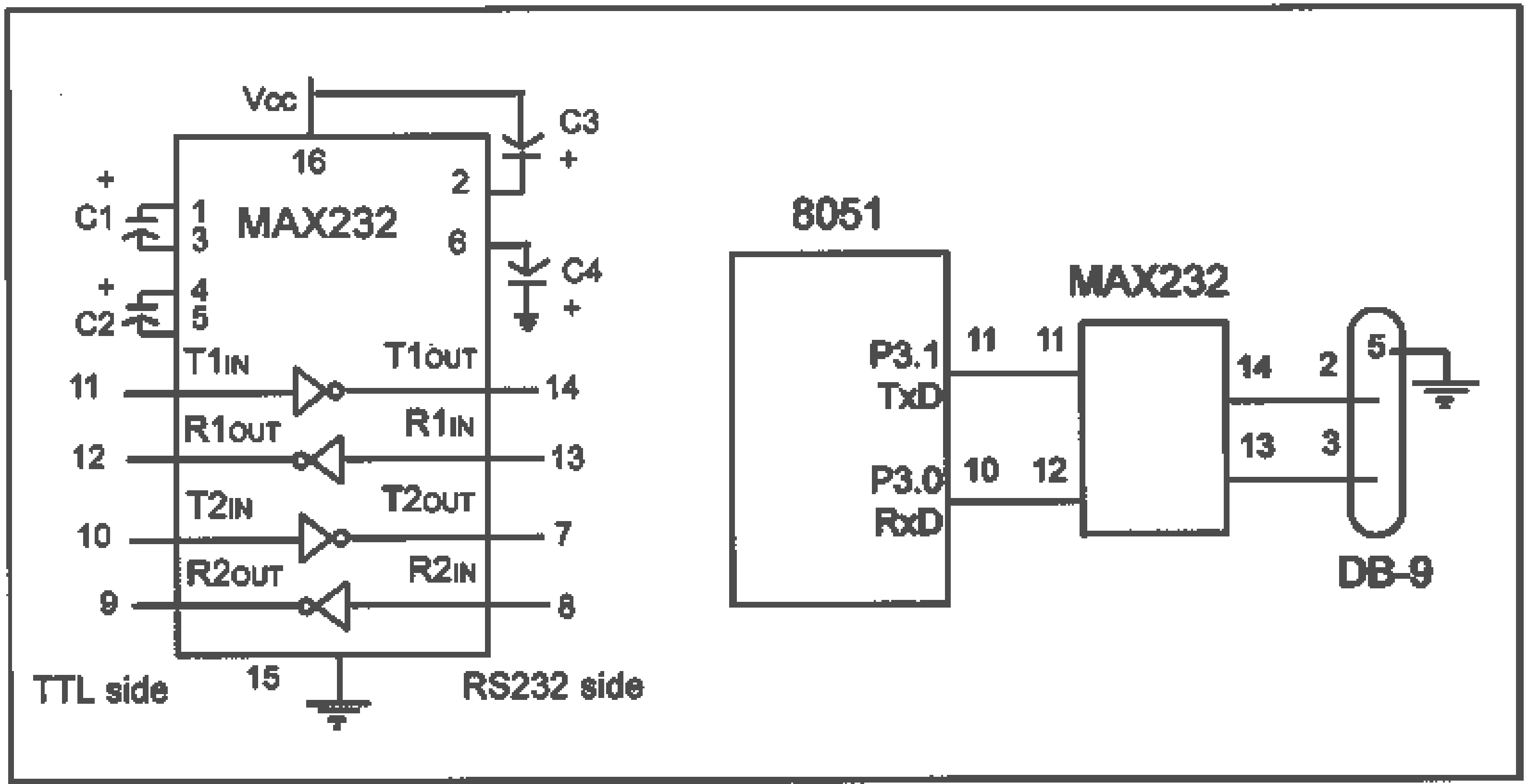


Figure 10.7: (a) Inside MAX232 and (b) its Connection to the 8051 (Null Modem)

Serial Interface

- The serial port of 8051 is full duplex, i.e., it can transmit and receive simultaneously.
- The register SBUF is used to hold the data.
- The special function register SBUF is physically two registers.
- One is, write-only and is used to hold data to be transmitted out of the 8051 via TXD.
- The other is, read-only and holds the received data from external sources via RXD.
- Both mutually exclusive registers have the same address 099H.

SBUF register

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

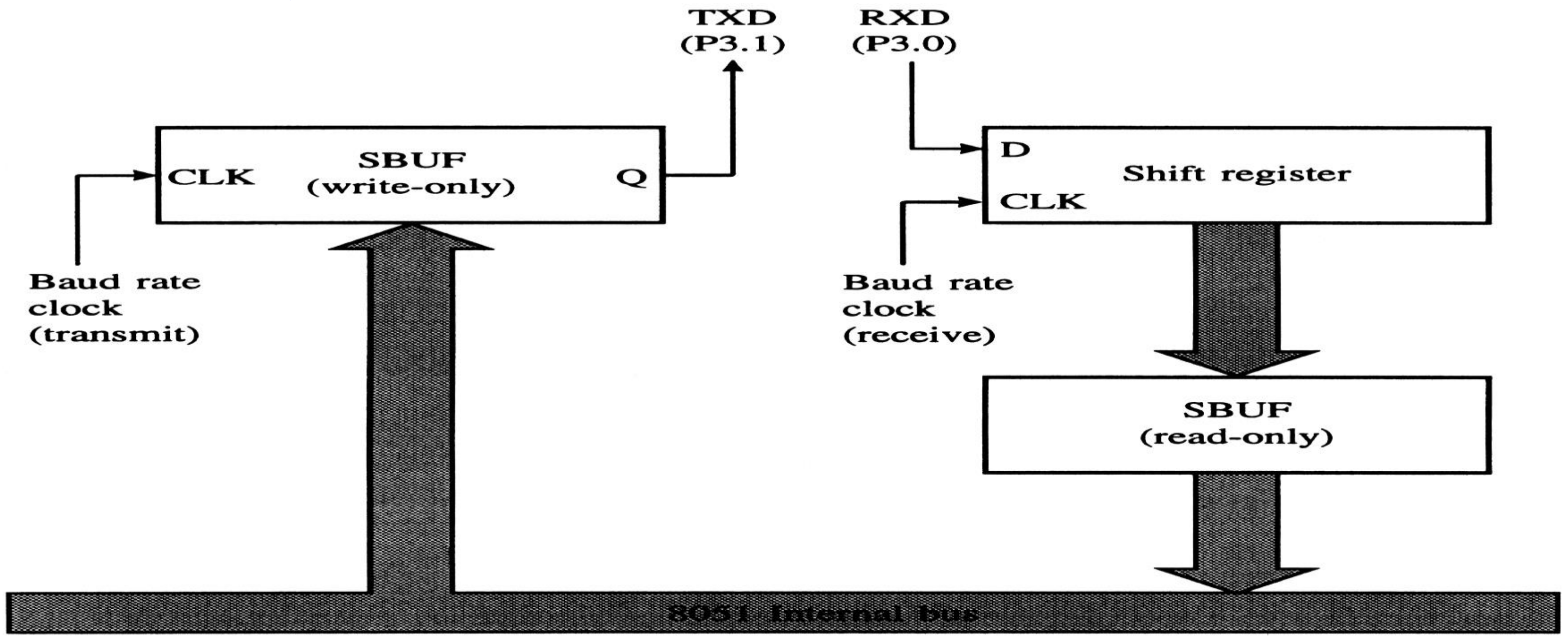


FIGURE 5—1
Serial port block diagram

Serial Interface

- **Serial Port Control Register (SCON)**
- Register SCON controls serial data communication. Address: 098H (Bit addressable)

7	6	5	4	3	2	1	0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

SM0 : mode specifier
SM1 : mode specifier

SM0	**SM1**	**MODE**	**operation**	**transmit rate**
0	**0**	**0**	**shift register**	**fixed (xtal/12)**
0	**1**	**1**	**8 bit UART**	**variable (timer1)**
1	**0**	**2**	**9 bit UART**	**fixed (xtal/32 or xtal/64)**
1	**1**	**3**	**9 bit UART**	**variable (timer1)**

SM2 : **used for multi processor communication**

REN : **receive enable (by software enable/disable)**

TB8 : **transmit bit8-used for serial mode 2 & 3**

RB8 : **receive bit 8-In serial mode 1, this bit gets a copy of the stop bit when an 8-bit data is received**

TI : **transmit interrupt flag set by HW after send , clear by SW-when the 8051 finishes 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 flag set by HW after received ,clear by SW**

- **When the 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in SBUF register**
- **Then it raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost**

- The next bit, **SM2**, is a flag for "Multiprocessor communication.
- Generally, whenever a byte has been received the 8051 will set the "RI" (Receive Interrupt) flag. This lets the program know that a byte has been received and that it needs to be processed.
- When SM2 is set the "RI" flag will only be triggered if the 9th bit received was a "1".

- The SCON SFR allows us to configure the Serial Port. Go through each bit and review its function.
- The first four bits (bits 4 through 7) are configuration bits.
- Bits **SM0** and **SM1** let us set the *serial mode* to a value between 0 and 3, inclusive.
- Selecting the Serial Mode selects the mode of operation (8-bit/9-bit, UART or Shift Register) and also determines how the baud rate will be calculated.
- In modes 0 and 2 the baud rate is fixed based on the oscillators frequency.
- In modes 1 and 3 the baud rate is variable based on how often Timer 1 overflows.

- That is to say, if SM2 is set and a byte is received whose 9th bit is clear, the RI flag will never be set.
- This can be useful in certain advanced serial applications.
- For now it is safe to say that you will almost always want to clear this bit so that the flag is set upon reception of *any* character

- The next bit, **REN**, is "Receiver Enable." This bit is very straightforward: If you want to receive data via the serial port, set this bit. You will almost always want to set this bit.
- The last four bits (bits 0 through 3) are operational bits. They are used when actually sending and receiving data--they are not used to configure the serial port.

- The **TB8** bit is used in modes 2 and 3. In modes 2 and 3, a total of nine data bits are transmitted.
- The first 8 data bits are the 8 bits of the main value, and the ninth bit is taken from TB8. If TB8 is set and a value is written to the serial port, the data bits will be written to the serial line followed by a "set" ninth bit. If TB8 is clear the ninth bit will be "clear."

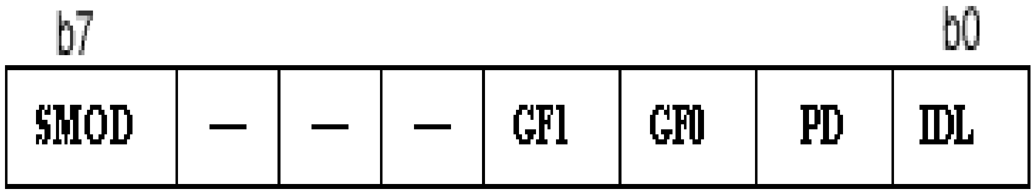
- The **RB8** also operates in modes 2 and 3 and functions essentially the same way as TB8, but on the reception side.
- When a byte is received in modes 2 or 3, a total of nine bits are received.
- In this case, the first eight bits received are the data of the serial byte received and the value of the ninth bit received will be placed in RB8.

- **TI** means "Transmit Interrupt." When a program writes a value to the serial port, certain amount of time will pass before the individual bits of the byte are "clocked out" the serial port.
- If the program were to write another byte to the serial port before the first byte was completely output, the data being sent would be garbled.

- Thus, the 8051 lets the program know that it has "clocked out" the last byte by setting the TI bit.
- When the TI bit is set, the program may assume that the serial port is "free" and ready to send the next byte.

- Finally, the **RI** bit means "Receive Interrupt." It functions similarly to the "TI" bit, but it indicates that a byte has been received.
- That is to say, whenever the 8051 has received a complete byte it will trigger the RI bit to let the program know that it needs to read the value quickly, before another byte is read.

Power Control Register(PCON)



b7							b0
SMOD	—	—	—	GF1	GF0	PD	IDL

- SMOD: Serial baud rate modify bit
  GF1: General purpose user flag bit 1
  GF0: General purpose user flag bit 0
  PD: Power down bit
  IDL: Idle mode bit

Serial Interface

- **Power Mode control Register**
- Register PCON controls processor power down, sleep modes and serial data baud rate. Only one bit of PCON is used with respect to serial communication. The seventh bit (b7) (SMOD) is used to generate the baud rate of serial communication.
- Address: 87H

Baud Rate

- baud: A data transmission rate (bits/second) for modems
- The Baud Rate is determined based on the oscillators frequency when in mode 0 and 2.
- In mode 0, the baud rate is always the oscillator frequency divided by 12.
- This means if crystal is 11.0592Mhz, mode 0 baud rate will always be 921,600 baud.
- In mode 2 the baud rate is always the oscillator frequency divided by 64, so a 11.0592Mhz crystal speed will yield a baud rate of 172,800.

- In modes 1 and 3, the baud rate is determined by how frequently timer 1 overflows.
- The more frequently timer 1 overflows, the higher the baud rate.
- There are many ways one can cause timer 1 to overflow at a rate that determines a baud rate, but the most common method is to put timer 1 in 8-bit auto-reload mode (timer mode 2) and set a reload value (TH1) that causes Timer 1 to overflow at a frequency appropriate to generate a baud rate.

Baud rate in the 8051

- 8051 transfers and receives data serially at many different baud rates
- The baud rates in the 8051 is programmable
- This is done with the help of timer 1
- 8051 divides the crystal frequency by 12 to get the machine cycle frequency
- In the case of XTAL=11.0592Mhz, the machine cycle frequency is 921.6khz
- The 8051’s serial communication UART circuitry divides the machine cycle frequency of 921.6khz by 32 once more before it is used by timer1 to set the baud rate

- To determine the value that must be placed in TH1 to generate a given baud rate, we may use the following equation (assuming PCON.7 is clear).

$$9600 = \frac{2^0}{32} \times \frac{11.0592 \times 10^6}{12 \times (256 - TH1)}$$

- TH1 = 256 - ((Crystal / 384) / Baud)
- If PCON.7(SMOD) is set then the baud rate is effectively doubled, thus the equation becomes:
- TH1 = 256 - ((Crystal / 192) / Baud)

- Therefore, 921.6khz /32 gives 28,800hz
- When timer 1 is used to set the baud rate it must be programmed in mode 2, ie 8 bit auto reload
- To get baud rates compatible with the PC, we must load TH1 with the values shown below

Baud rate	TH1(decimal)	TH1(hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

- XTAL=11.0592Mhz

Example

- With XTAL=11.0592 MHz, find the TH1 value needed to have the following baud rate.

(a) 9600

(b) 2400

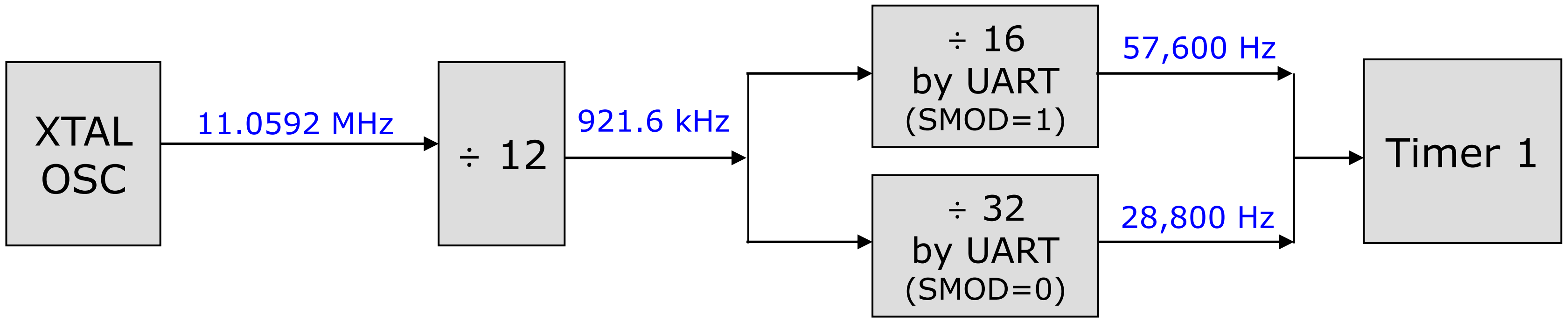
(c) 1200

Solution:with XTAL=11.0592 MHz:

Machine cycle frequency: 11.0592 MHz/12 = 921.6kHz,
Frequency provided by UART: 921.6kHz/32 =28,800 Hz (if SMOD=0)
or 921.6kHz/16 =57,600 Hz (if SMOD=1)
should be provided by Timer 1 to set baud rate.

Case I: SMOD=0.
(a) 28,800/9,600 = 3. 256-3 = 253 = FDh
(b) 28,800/2,400 = 12. 256-12 = 244 = F4h
(c) 28,800/1,200 = 24. 256-24 = 232 = E8h
should be loaded into TH1.

Case II: SMOD=1.
(a) 57,600/9,600 = 6. 256-6 = 250 = FAh
(b) 57,600/2,400 = 24. 256-24 = 232 = E8h
(c) 57,600/1,200 = 48. 256-48 = 232 = D0h
should be loaded into TH1.



Programming the 8051 to transfer the data serially

1. TMOD register is loaded with the value 20h, indicating the use of timer 1 in mode2 to set the baud rate
2. The TH1 is loaded with one of the values (refer table) to set the baud rate for serial data transfer(assuming XTAL=11.0592mhz)
3. The SCON register is loaded with the value 50h, indicating serial mode1, where an 8-bit data is framed with start and stop bits
4. TR1 is set to start timer1
5. TI is cleared by the CLR TI instruction

6. The character byte to be transferred serially is written into the SBUF register.
7. The TI flag bit is monitored using the instruction JNB TI, xx to see if the character has been transferred completely.
8. To transfer the next character, goto step 5

- Write a program for the 8051 to transfer letter "A" serially at 4800 baud, continuously.

Solution:

```
       MOV    TMOD,#20H     ;timer 1, mode 2 (auto reload)
       MOV    TH1,#-6       ;4800 baud rate
       MOV    SCON,#50H     ;8-bit, 1 stop, REN enabled
       SETB   TR1           ;start timer 1
AGN:   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   AGN           ;keep sending A
```

Data Transmission

- **Data Transmission** :Transmission of serial data begins at any time when data is written to SBUF.
- Pin P3.1 (Alternate function bit TXD) is used to transmit data to the serial data network.
- TI is set to 1 when data has been transmitted. This signifies that SBUF is empty so that another byte can be sent

Data Reception

- Reception of serial data begins if the receive enable bit is set to 1 for all modes. Pin P3.0 (Alternate function bit RXD) is used to receive data from the serial data network.
- Receive interrupt flag, RI, is set after the data has been received in all modes. The data gets stored in SBUF register from where it can be read

Reading the Serial Port or Data Reception

- Reading data received by the serial port is equally easy.
- To read a byte from the serial port one just needs to read the value stored in the **SBUF** (99h) SFR after the 8051 has automatically set the **RI** flag in SCON.

- For example, if your program wants to wait for a character to be received and subsequently read it into the Accumulator, the following code segment may be used
- **JNB RI,$** ;Wait for the 8051 to set the RI flag **MOV A,SBUF** ;Read the character from the serial port

- The first line of the above code segment waits for the 8051 to set the RI flag; again, the 8051 sets the RI flag automatically when it receives a character via the serial port.
- So as long as the bit is not set the program repeats the "JNB" instruction continuously.
- Once the RI bit is set upon character reception the above condition automatically fails and program flow falls through to the "MOV" instruction which reads the value.

Importance of TI flag

- Ex: to transmit a character via TxD-steps

1. The byte character to be transmitted is written into the SBUF register
2. The start bit is transferred
3. The 8-bit character is transferred one bit at a time
4. The stop bit is transferred. It is during the transfer of the stop bit that the 8051 raises th TI flag indicating that the last character was transmitted and it is ready to transfer the next character

5. By monitoring the TI flag, we make sure that we are nor overloading the SBUF register

- If we write another byte into the SBUF before TI is raised, the untransmitted portion of the previous byte will be lost. In other words, when the 8051 finishes transferring a byte, it raises the TI flag to indicate it is ready for the next character

6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by the CLR TI instruction in order for this new byte to be transferred

8051 INTERRUPTS

- 8051 provides 5 vectored interrupts.
- TF0
- TF1
- RI/TI
- INT0
- INT1
- Out of these, INT0 and INT1 are external interrupts whereas Timer and Serial port interrupts are generated internally. The external interrupts could be negative edge triggered or low level triggered.