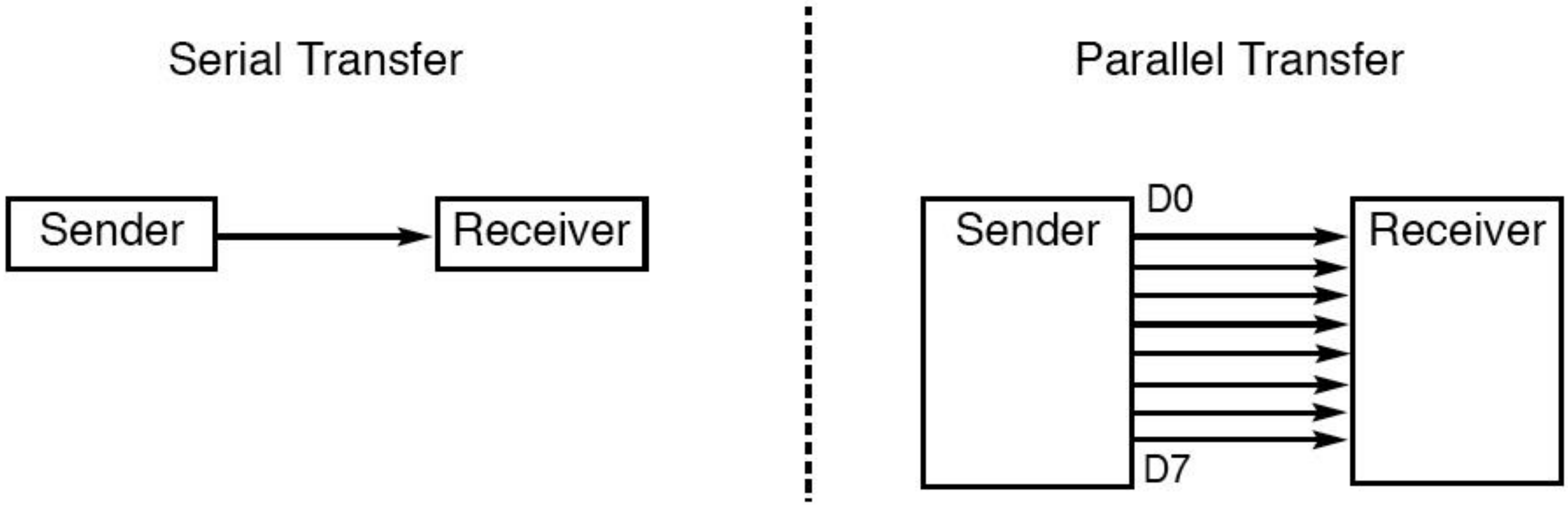
The 8051 Microcontroller and Embedded Systems

8051 SERIAL PORT PROGRAMMING

OBJECTIVES

- Contrast and compare serial versus parallel communication
- List the advantages of serial communication over parallel
- Explain serial communication protocol
- Contrast synchronous versus asynchronous communication
- Contrast half-versus full-duplex transmission
- Explain the process of data framing
- Describe data transfer rate and bps rate
- Define the RS232 standard
- Explain the use of the MAX232 and MAX233 chips

BASICS OF SERIAL COMMUNICATION



Serial versus Parallel Data Transfer

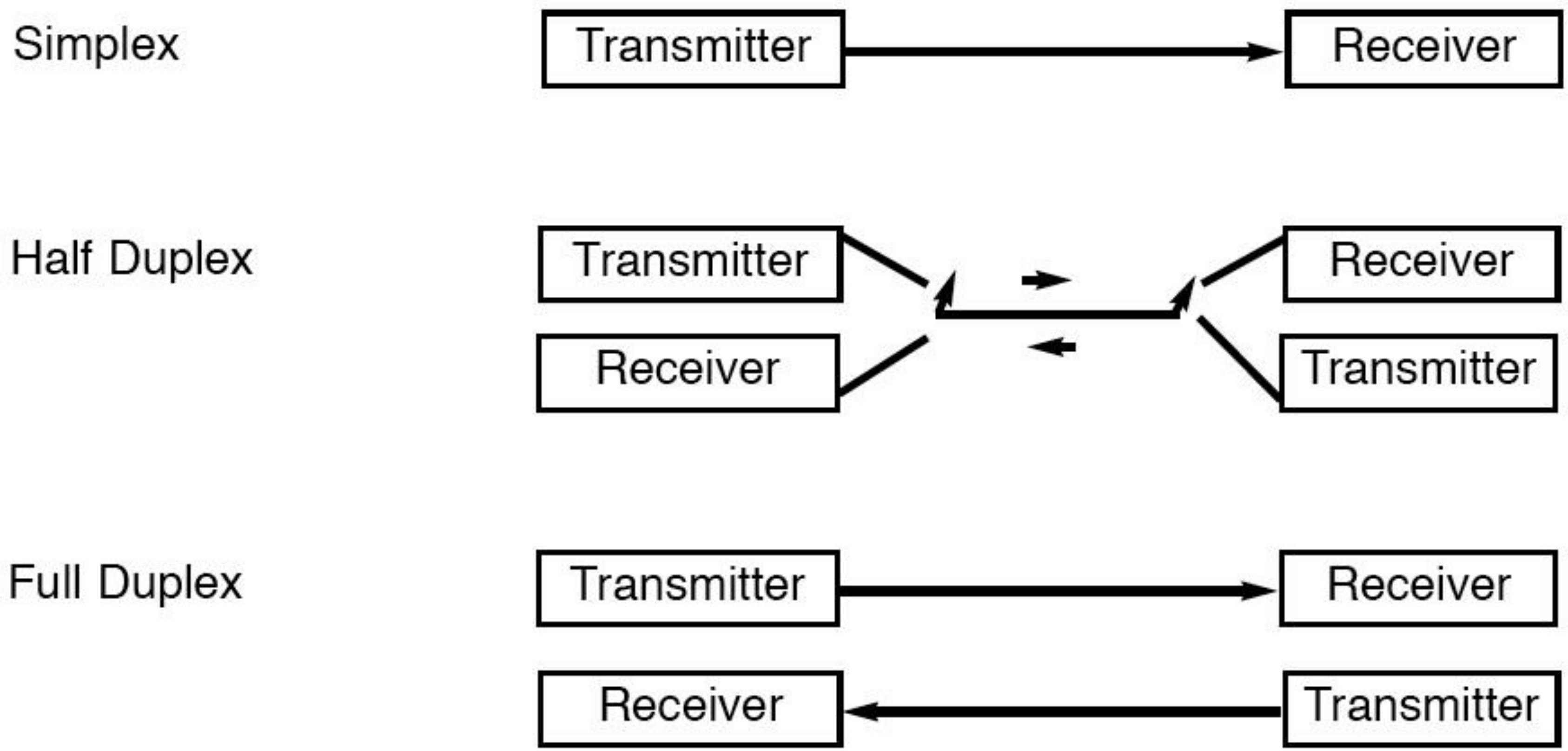
BASICS OF SERIAL COMMUNICATION

- serial communication uses single data line making it much cheaper
- enables two computers in different cities to communicate over the telephone
- byte of data must be converted to serial bits using a parallel-in-serial-out shift register and transmitted over a single data line
- receiving end there must be a serial-in-parallel-out shift register
- if transferred on the telephone line, it must be converted to audio tones by *modem*
- for short distance the signal can be transferred using wire
- how PC keyboards transfer data to the motherboard

BASICS OF SERIAL COMMUNICATION

- Two methods, asynchronous and synchronous
  - *synchronous* method transfers a block of data (characters) at a time
  - *asynchronous* method transfers a single byte at a time
- Uses special IC chips called
  - **UART** (universal asynchronous receiver-transmitter) and
  - **USART** (universal synchronousasynchronous receiver-transmitter)
- 8051 chip has a built-in UART

BASICS OF SERIAL COMMUNICATION



Simplex, Half-, and Full-Duplex Transfers

BASICS OF SERIAL COMMUNICATION

- Half- and full-duplex transmission
  - if the data can be transmitted and received, it is a *duplex* transmission
  - *simplex* transmissions the computer only sends data
  - duplex transmissions can be half or full duplex
  - depends on whether or not the data transfer can be simultaneous
  - If one way at a time, it is *half duplex*
  - If can go both ways at the same time, it is full duplex
  - full duplex requires two wire conductors for the data lines (in addition to the signal ground)

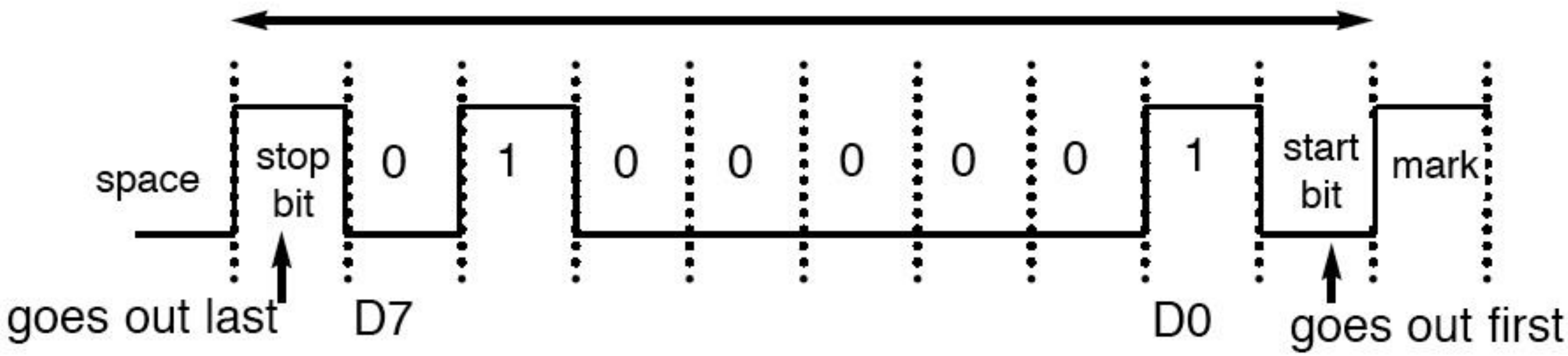
BASICS OF SERIAL COMMUNICATION

- **Asynchronous serial communication and data framing**
  - **data coming in 0s and 1s**
  - **to make sense of the data sender and receiver agree on a set of rules**
  - **Protocol**
    - **how the data is packed**
    - **how many bits/character**
    - **when the data begins and ends**

BASICS OF SERIAL COMMUNICATION

- **Start and stop bits**
  - asynchronous method, each character is placed between start and stop bits
  - called *framing*
  - start bit is always one bit
  - stop bit can be one or two bits
  - start bit is always a 0 (low)
  - stop bit(s) is 1 (high)
  - LSB is sent out first

BASICS OF SERIAL COMMUNICATION



Framing ASCII “A” (41H)

BASICS OF SERIAL COMMUNICATION

- **in modern PCs one stop bit is standard**
- **when transferring a text file of ASCII characters using 1 stop bit there is total of 10 bits for each character**
- **8 bits for the ASCII code (1 parity bit), 1 bit each for the start and stop bits**
- **for each 8-bit character there are an extra 2 bits, which gives 20% overhead**

BASICS OF SERIAL COMMUNICATION

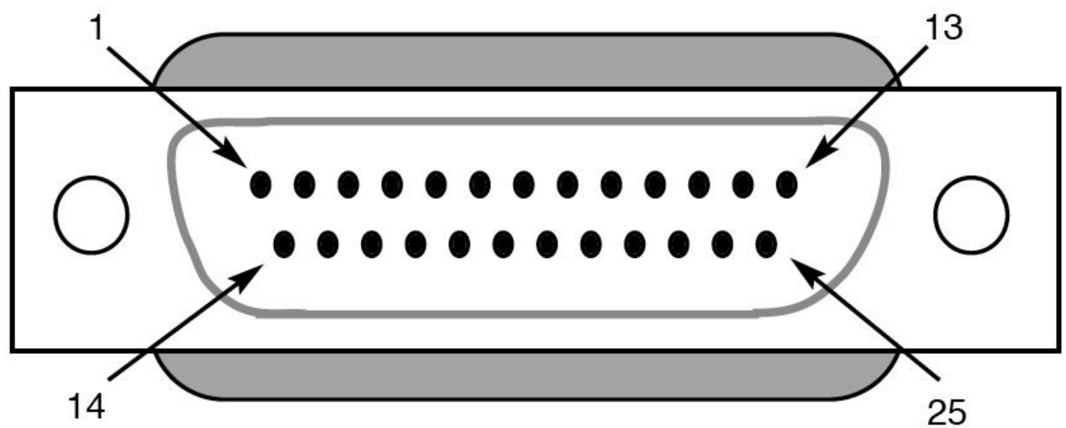
- **Data transfer rate**
  - **rate of data transfer *bps* (bits per second)**
  - **widely used terminology for bps is *baud rate***
  - **baud and bps rates are not necessarily equal**
  - **baud rate is defined as the number of signal changes per second**

BASICS OF SERIAL COMMUNICATION

- **RS232 standards**
  - most widely used serial I/O interfacing standard
  - input and output voltage levels are not TTL compatible
  - 1 bit is represented by -3 to -25 V
  - 0 bit is +3 to +25 V
  - -3 to +3 is undefined
  - to connect RS232 to a microcontroller system must use voltage converters such as MAX232 to convert the TTL logic levels to the RS232 voltage levels, and vice versa
  - MAX232 IC chips are commonly referred to as line drivers

BASICS OF SERIAL COMMUNICATION

Pin	Description
1	Protective ground
2	Transmitted data (TxD)
3	Received data (RxD)
4	Request to send ($\overline{\text{RTS}}$)
5	Clear to send ($\overline{\text{CTS}}$)
6	Data set ready ($\overline{\text{DSR}}$)
7	Signal ground (GND)
8	Data carrier detect ($\overline{\text{DCD}}$)
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 ($\overline{\text{DTR}}$)
21	Signal quality detector
22	Ring indicator
23	Data signal rate select
24	Transmit signal element timing
25	Unassigned



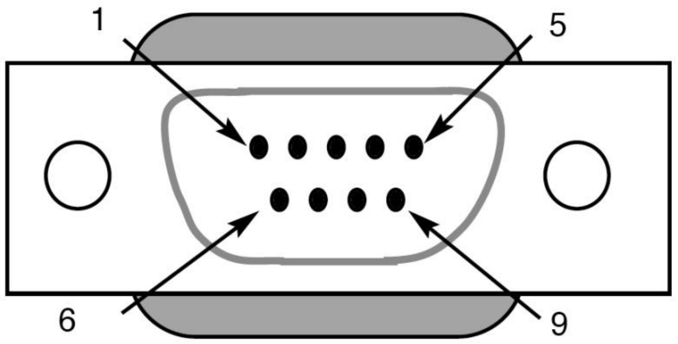
RS232 Connector DB-25

RS232 Pins (DB-25)

BASICS OF SERIAL COMMUNICATION

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

IBM PC DB-9 Signals

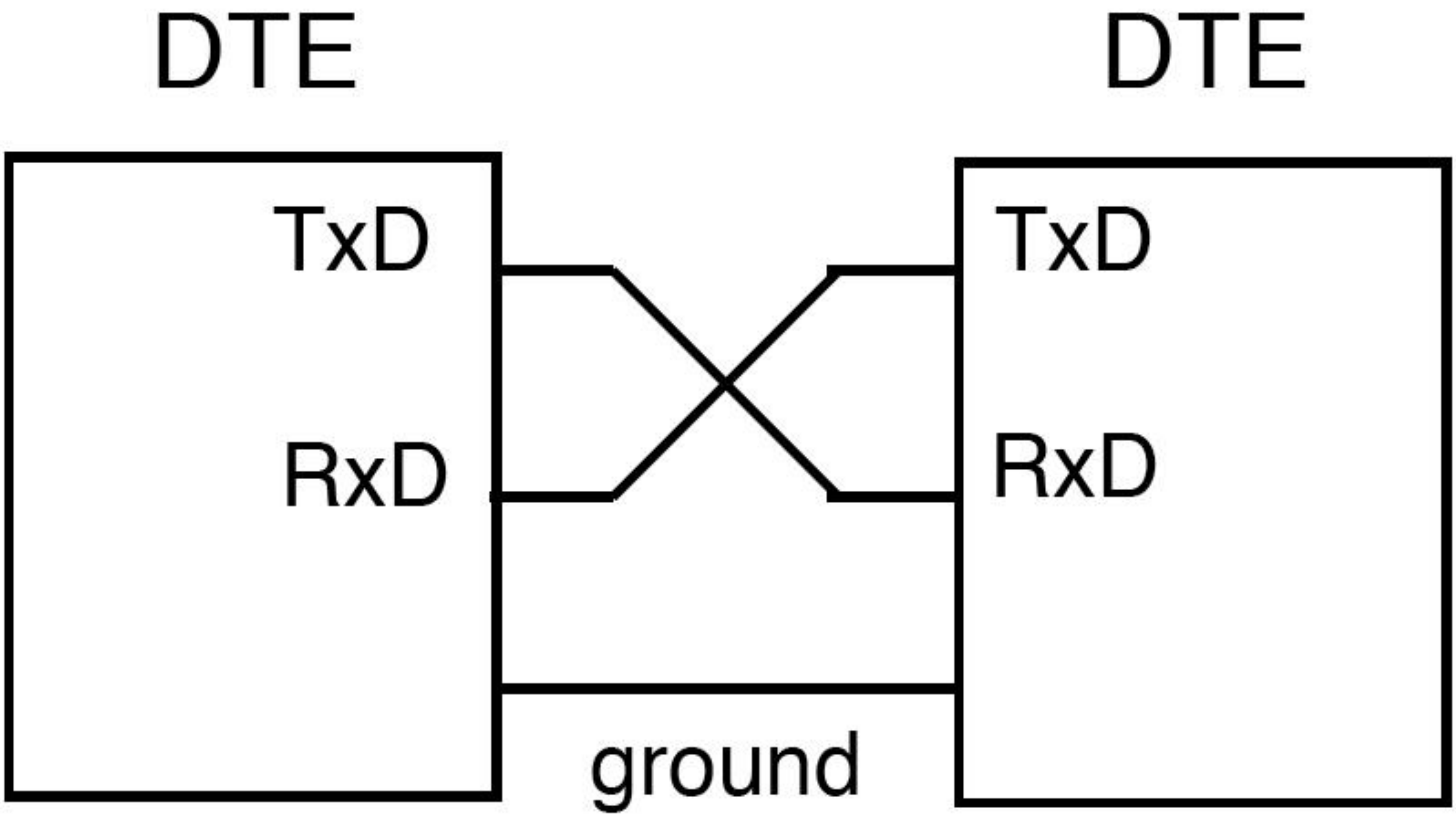


DB-9 9-Pin Connector

BASICS OF SERIAL COMMUNICATION

- **Data communication classification**
  - **DTE (data terminal equipment)**
  - **DCE (data communication equipment)**
  - **DTE - terminals and computers that send and receive data**
  - **DCE - communication equipment responsible for transferring the data**
  - **simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground**

BASICS OF SERIAL COMMUNICATION



Null Modem Connection

BASICS OF SERIAL COMMUNICATION

- **Examining RS232 handshaking signals**
  - **many of the pins of the RS-232 connector are used for handshaking signals**
  - **they are not supported by the 8051 UART chip**

BASICS OF SERIAL COMMUNICATION

- **PC/compatible COM ports**
  - **PC/compatible computers (Pentium) microprocessors normally have two COM ports**
  - **both ports have RS232-type connectors**
  - **COM ports are designated as COM 1 and COM 2 (replaced by USB ports)**
  - **can connect the 8051 serial port to the COM 2 port**

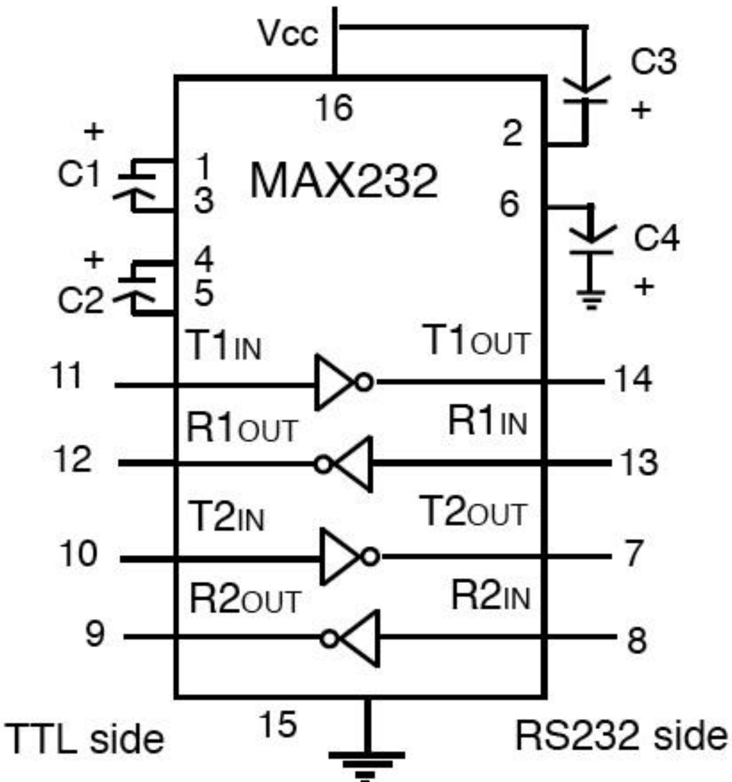
8051 CONNECTION TO RS232

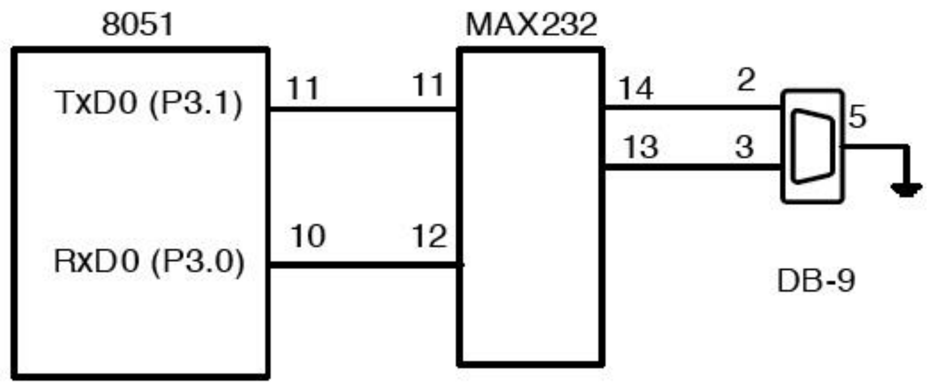
- **RxD and TxD pins in the 8051**
  - 8051 has two pins used for transferring and receiving data serially
  - TxD and RxD are part of the port 3 group
  - pin 11 (P3.1) is assigned to TxD
  - pin 10 (P3.0) is designated as RxD
  - these pins are TTL compatible
  - require a line driver to make them RS232 compatible
  - driver is the MAX232 chip

8051 CONNECTION TO RS232

- **MAX232**
  - converts from RS232 voltage levels to TTL voltage levels
  - uses a +5 V power source
  - MAX232 has two sets of line drivers for transferring and receiving data
  - line drivers used for TxD are called T1 and T2
  - line drivers for RxD are designated as R1 and R2
  - T1 and R1 are used together for TxD and RxD of the 8051
  - second set is left unused

8051 CONNECTION TO RS232



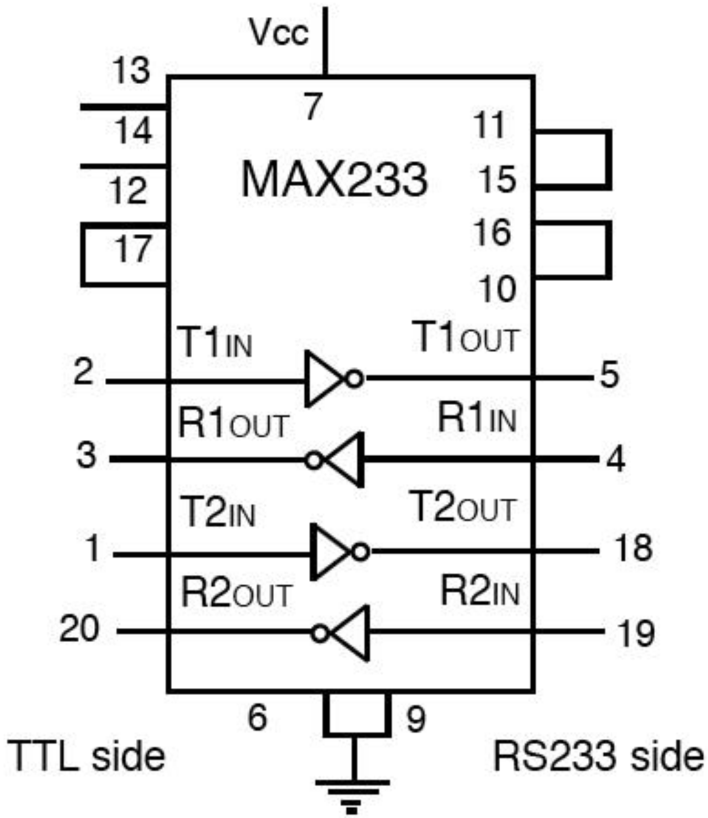


(a) Inside MAX232
(b) its Connection to the 8051 (Null Modem)

8051 CONNECTION TO RS232

- **MAX233**
  - **MAX233 performs the same job as the MAX232**
  - **eliminates the need for capacitors**
  - **much more expensive than the MAX232**

8051 CONNECTION TO RS232



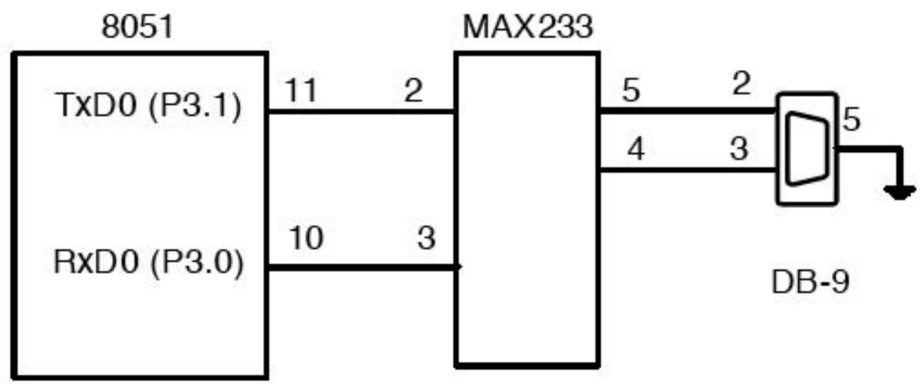


Figure 10–8 (a) Inside MAX233
(b) Its Connection to the 8051 (Null Modem)

8051 SERIAL PORT PROGRAMMING

- **Baud rate in the 8051**
  - **serial communications of the 8051 with the COM port of the PC**
  - **must make sure that the baud rate of the 8051 system matches the baud rate of the PC's COM port**
  - **can use Windows HyperTerminal program**

8051 SERIAL PORT PROGRAMMING

110
150
300
600
1200
2400
4800
9600
19200

Note: Some of the Baud rates supported by 486/ Pentium IBM PC BIOS.

PC Baud Rates

8051 SERIAL PORT PROGRAMMING

- **Baud rate in the 8051**
  - baud rate in the 8051 is programmable
  - done with the help of Timer 1
  - relationship between the crystal frequency and the baud rate in the 8051
  - 8051 divides the crystal frequency by 12 to get the machine cycle frequency
  - 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 must be programmed in mode 2, that is 8-bit, auto-reload

8051 SERIAL PORT PROGRAMMING

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

Note: XTAL = 11.0592 MHz.

Timer 1 TH1 Register Values for Various Baud Rates

Example 10-1
With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates.
(a) 9600 (b) 2400 (c) 1200

- **machine cycle frequency**
  = 11.0592 MHz / 12 = 921.6 kHz
- **Timer 1 frequency provided by 8051 UART**
  = 921.6 kHz / 32 = 28,800 Hz

(a) 28,800 / 3 = 9600	**where -3**	**= FD (hex)**
(b) 28,800 / 12 = 2400	**where -12**	**= F4 (hex)**
(c) 28,800 / 24 = 1200	**where -24**	**= E8 (hex)**

8051 SERIAL PORT PROGRAMMING

- **SBUF (serial buffer) register**
  - **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**
  - **can be accessed like any other register**

    MOV SBUF,#'D'   ;load SBUF=44H, ASCII for 'D'

    MOV SBUF,A   ;copy accumulator into SBUF

    MOV A,SBUF   ;copy SBUF into accumulator
  - **when a byte is written, it is framed with the start and stop bits and transferred serially via the TxD pin**
  - **when the bits are received serially via RxD, it is deframe by eliminating the stop and start bits, making a byte out of the data received, and then placing it in the SBUF**

8051 SERIAL PORT PROGRAMMING

- **SCON (serial control) register**
  - **to program the start bit, stop bit, and data bits**

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.

SCON Serial Port Control Register (Bit-Addressable)

8051 SERIAL PORT PROGRAMMING

- SM0 and SM1 determine the mode
- only mode 1 is important
- when mode 1 is chosen, the data framing is 8 bits, 1 stop bit, and 1 start bit
- compatible with the COM port of PCs
- mode 1 allows the baud rate to be variable and is set by Timer 1 of the 8051
- for each character a total of 10 bits are transferred, where the first bit is the start bit, followed by 8 bits of data, and finally 1 stop bit.

8051 SERIAL PORT PROGRAMMING

- REN (receive enable)
- REN=1, allows 8051 to receive data on the RxD
- if 8051 is to both transfer and receive data, REN must be set to 1
- REN=0, the receiver is disabled
- SETB SCON.4 and CLR SCON.4,

8051 SERIAL PORT PROGRAMMING

- TI (transmit interrupt)
  - when 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)
  - when the 8051 receives data serially via RxD, it places the byte in the SBUF register
  - then raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost

8051 SERIAL PORT PROGRAMMING

- **Program to transfer data serially**
  1. **TMOD register is loaded with the value 20H**
  2. **TH1 is loaded with value to set the baud rate**
  3. **SCON register is loaded with the value 50H**
  4. **TR1 is set to 1 to start Timer1**
  5. **TI is cleared by the "CLR TI" instruction**
  6. **transmit character byte is written into the SBUF register**
  7. **TI flag bit is monitored to see if the character has been transferred completely**
  8. **to transfer the next character, go to Step 5.**

Example 10-2

Write a program to transfer letter "A" serially at 4800 baud, continuously. (Error in line 3 – should be SCON)

```
#include<reg51.h>
Void main()
{
   TMOD = 0x20;
   TH1= 0xFD ;
   SCON = 0x50;
   TR1 = 1;
   while(1){
         SBUF = 'A' ;
         While (T1= =0 );
         T1 = 0;
   }
}
```

```
01 MOV TMOD,#20H          ;Timer 1, mode 2(auto-reload)
02 MOV TH1,#-6            ;4800 baud rate
03 MOV TCON,#50H          ;8-bit, 1 stop, REN enabled
04 SETB TR1               ;start Timer 1
05 AGAIN: MOV SBUF,#"A"   ;letter "A" to be transferred
06 HERE: JNB TI,HERE      ;wait for the last bit
07 CLR TI                 ;clear TI for next char
08 SJMP AGAIN             ;keep sending A
09
10 END
```

Example 10-3

Write a program to transfer the message "YES" serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

(Error in line 3 – should be SCON)

```
01 MOV TMOD,#20H            ;Timer 1 Mode 2
02 MOV TH1,#-3              ;9600 baud
03 MOV TCON,#50H            ;8-bit, 1 stop bit, REN enabled
04 SETB TR1                 ;start Timer 1
05 AGAIN: MOV A,#"Y"        ;transfer "Y"
06 ACALL TRANS
07 MOV A,#"E"               ;transfer "E"
08 ACALL TRANS
09 MOV A, #"S"              ;transfer "S"
10 ACALL TRANS
11 SJMP AGAIN               ;keep doing it
12 ;--------------------------------
13 ;serial data transfer subroutine
14 ;--------------------------------
15 TRANS: MOV SBUF,A        ;load SBUF
16 HERE: JNB TI,HERE        ;wait for last bit to transfer
17 CLR TI                   ;get ready for next byte
18 RET
19
20 END
```

SECTION 10.3: 8051 SERIAL PORT PROGRAMMING IN ASSEMBLY

- **Importance of the TI flag**
  - **check the TI flag bit, we know whether can transfer another byte**
  - **TI flag bit is raised by the 8051**
  - **TI flag cleared by the programmer**
  - **writing a byte into SBUF before the TI flag bit is raised, may lead to loss of a portion of the byte being transferred**

SECTION 10.3: 8051 SERIAL PORT PROGRAMMING IN ASSEMBLY

- **Program to receive data serially**
  1. **TMOD register is loaded with the value 20H**
  2. **TH1 is loaded with value set the baud rate**
  3. **SCON register is loaded with the value 50H**
  4. **TR1 is set to 1 to start Timer 1**
  5. **RI is cleared with the "CLR RI" instruction**
  6. **RI flag bit is monitored to see if an entire character has been received yet**
  7. **RI=1 SBUF has the byte, its contents are moved into a safe place**
  8. **to receive the next character, go to Step 5**

Example 10-4

Program 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.

(Error in line 3 – should be SCON)

```
01 MOV TMOD,#20H          ;Timer 1, mode 2 (auto reload)
02 MOV TH1,#-6            ;4800 baud
03 MOV TCON,#50H          ;8-bit, 1 stop, REN enabled
04 SETB TR1               ;start Timer 1
05 HERE: JNB RI,HERE      ;wait for char to come in
06 MOV A,SBUF             ;save incoming byte in A
07 MOV P1,A               ;send to port 1
08 CLR RI                 ;get ready to receive next byte
09 SJMP HERE              ;keep getting data
10
11 END
```

SECTION 10.3: 8051 SERIAL PORT PROGRAMMING IN ASSEMBLY

- **Importance of the RI flag bit**
  1. **it receives the start bit, next bit is the first bit of the character**
  2. **when the last bit is received, a byte is formed and placed in SBUF**
  3. **when stop bit is received, makes RI = 1**
  4. **when RI=1, received byte is in the SBUF register, copy SBUF contents to a safe place**
  5. **after the SBUF contents are copied the RI flag bit must be cleared to 0**

SECTION 10.3: 8051 SERIAL PORT PROGRAMMING IN ASSEMBLY

- **Doubling the baud rate in the 8051**
  - **two ways to increase the baud rate**
    1. **Use a higher-frequency crystal**
    2. **Change a bit in the PCON register**

D7							D0
SMOD	--	--	--	GF1	GF0	PD	IDL

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.

Table 10–5 Baud Rate Comparison for SMOD = 0 and SMOD = 1

SECTION 10.3: 8051 SERIAL PORT PROGRAMMING IN ASSEMBLY

- **Interrupt-based data transfer**
  - **it is a waste of the microcontroller's time to poll the TI and RI flags**
  - **to avoid wasting time, use interrupts instead of polling**

Next …

- <u>Lecture Problems Textbook Chapter 10</u>
  - Answer as many questions as you can and submit via MeL before the end of the lecture.

- <u>Proteus Exercise 9</u>
  - Do as much of the Proteus exercise as you can and submit via MeL before the end of the lecture.