Dec Hex Bin
17 11 00010001

ORG ; SEVENTEEN

SERIAL PORT PROGRAMMING WITH ASSEMBLY AND C#

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall

OBJECTIVES

this chapter enables the student to:

- List the advantages of serial communication over parallel communication.
- Explain the difference between synchronous and asynchronous communication.
- Define the terms simplex, half duplex, and full duplex and diagram their implementation in serial communication.
- Describe how start and stop bits frame data for serial communication.

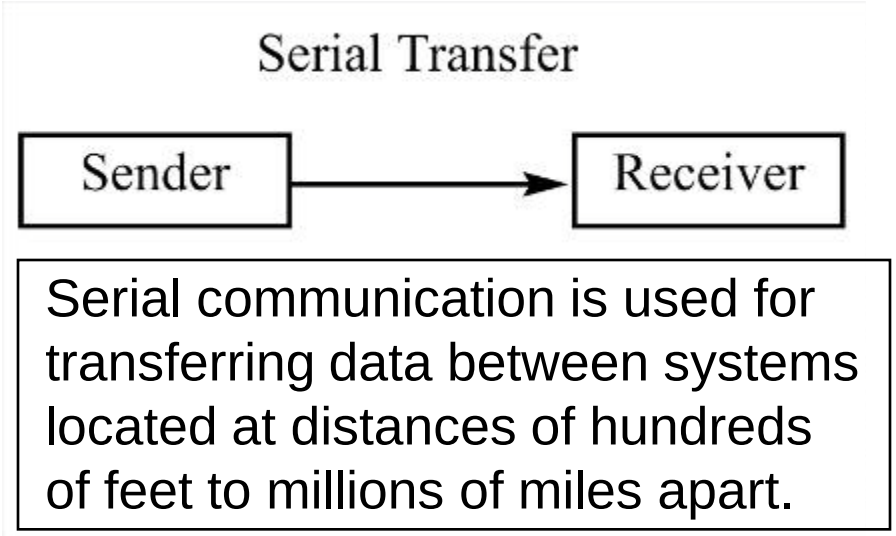
OBJECTIVES (*cont*)

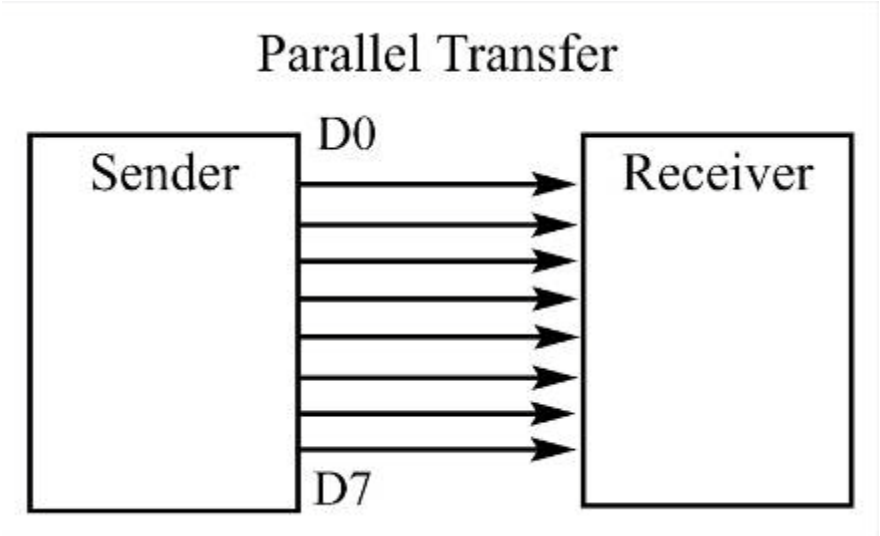
this chapter enables the student to:

- Contrast and compare the measures baud rate and bps. (bits per second)
- Describe the RS232 standard.
- Contrast and compare DTE (data terminal) versus DCE (data communication) equipment.
- Describe the purpose of handshaking signals such as DTR, RTS, and CTS.
- Code Assembly language instructions to perform serial communication using BIOS INT 14H.
- Describe the function of UART and USART chips .

17.1: BASICS OF SERIAL COMMUNICATION

- A microprocessor communicates with the outside world by providing data in byte-sized chunks.
  - Devices such as printers, grab data from the 8-bit data bus, and present to the 8-bit printer data bus.
    - An 8-Bit data path is expensive; signals can be diminished or distorted by cable length.





17.1: BASICS OF SERIAL COMMUNICATION

- As serial communication uses a single data line, it is possible for two computers located in two different cities to communicate over the telephone.
  - It is also much cheaper.
- Outgoing data must be grabbed from the 8-bit data bus of the microprocessor, and converted to serial bits, using a *parallel-in-serial-out* shift register.
  - The receiving end has a *serial-in-parallel-out* shift register.
    - To receive serial data, pack it into a byte, and present it to the system at the receiving end.

17.1: BASICS OF SERIAL COMMUNICATION

- Data transferred on the telephone line must be converted from 0s and 1s to audio tones.
  - Sinosoidal-shaped signals.
- This conversion is performed by a modem.
  - Short for "modulator/demodulator."

17.1: BASICS OF SERIAL COMMUNICATION

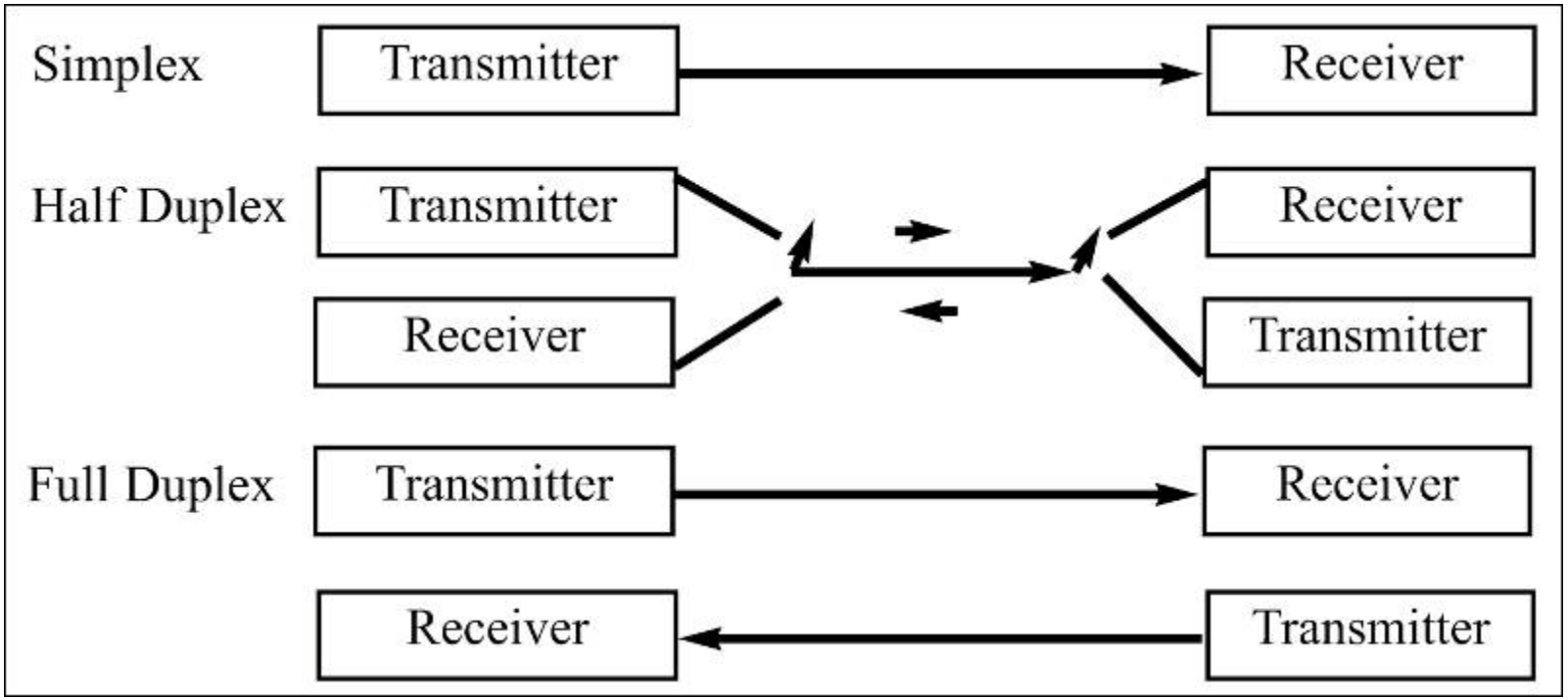
- Serial data communication uses two methods:
  - **Synchronous** - transfers a block of data (characters) at a time.
  - **Asynchronous** - transfers a single byte at a time.
- Special IC chips are made by many manufacturers for serial data communications, commonly referred
  - **UART** (universal asynchronous receiver-transmitter
  - **USART** (universal synchronous-asynchronous receiver-transmitter).
    - The x86 COM port uses the UART.

17.1: BASICS OF SERIAL COMMUNICATION
half- and full-duplex transmission

- During simplex transmissions, such as with printers, the computer only *sends* data.
- A *duplex* transmission is one in which the data can be transmitted *and* received.

17.1: BASICS OF SERIAL COMMUNICATION
half- and full-duplex transmission

- Duplex transmissions can be *half*- or *full*-duplex.
  - Data transmitted one way at a time is called *half duplex*.
  - Data going both ways at the same time is *full duplex*.



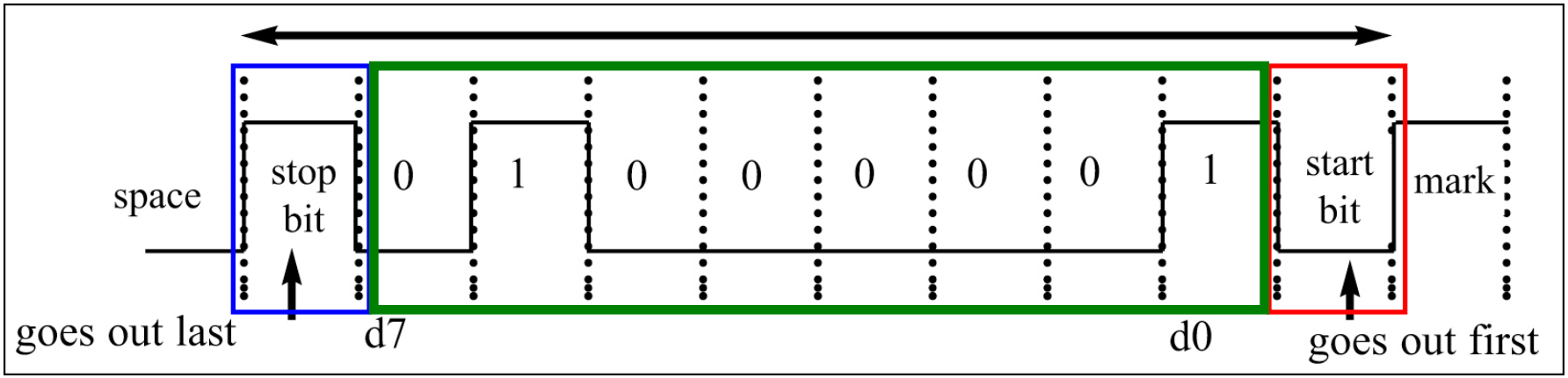
17.1: BASICS OF SERIAL COMMUNICATION protocol/usage

- Incoming data at the receiving end of the data line in a serial data transfer is all 0s and 1s.
  - To make sense of the data, sender and receiver agree on a set of rules, a *protocol*…
    - How the data is packed.
    - How many bits constitute a character.
    - When the data begins and ends.
- Asynchronous serial data communication is widely used for character-oriented transmissions.
  - Block-oriented data transfers use synchronous transfer.

17.1: BASICS OF SERIAL COMMUNICATION
data framing – start and stop bits

- In asynchronous communications, the **data**, such as ASCII characters, are packed between a start bit and a stop bit, a process called *framing.*

ASCII character "**A**", binary **0100 0001**, framed between the **start** bit and 2 stop bits.



- The **start** bit is always one bit and always a 0. (*low*)
- The **stop** bit can be one or two bits, and is 1 (*high*).

Fig. 17-1 Framing ASCII "A" (41H)

17.1: BASICS OF SERIAL COMMUNICATION
data framing – start and stop bits

- Asynchronous peripheral chips and modems can be programmed for data 5, 6, 7, or 8 bits wide.
  - In addition to the number of stop bits, 1 or 2.
- When there is no transfer, the signal is 1 (*high*).
  - Referred to as ***mark***.
  - The 0 (*low*) is referred to as ***space***.

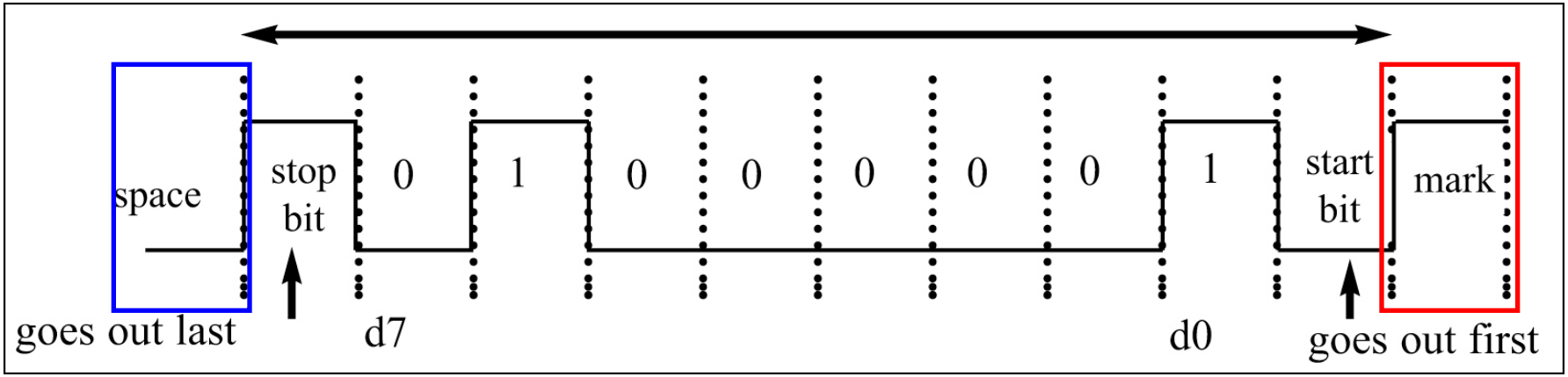


Fig. 17-1 Framing ASCII "A" (41H)

17.1: BASICS OF SERIAL COMMUNICATION
data framing – start and stop bits

- Transmission begins with a start bit, followed by D0, the LSB, then the rest of the bits until the MSB (D7)
  - Last, 2 stop bits indicating the end of the character "A".
- Asynchronous peripheral chips and modems can be programmed for data 5, 6, 7, or 8 bits wide.
  - In addition to the number of stop bits, 1 or 2.
- In some older systems, due to slowness of receiving mechanical devices, 2 stop bits were used to give the device time to organize before the next byte.
  - In modern PCs the use of 1 stop bit is common.

17.1: BASICS OF SERIAL COMMUNICATION
parity bit

- In some systems, for data integrity, the parity bit of the character byte is included in the data frame.
  - If a system requires it, the parity bit is transmitted after the MSB, and is followed by the stop bit.
    - An odd-parity bit the number of data bits, including the parity bit, has an odd number of 1s.
    - In an even-parity bit, the total number of bits, including the parity bit, is even.
- UART chips allow programming of the parity bit.
  - For odd-, even-, and no-parity options.

17.1: BASICS OF SERIAL COMMUNICATION data transfer rate

Example 17-1

Calculate the total number of bits used in transferring 50 pages, each with 80 × 25 characters. Assume 8 bits per character and 1 stop bit.

Solution:

For each character a total of 10 bits is used, 8 bits for the character, 1 stop bit, and 1 start bit. Therefore, the total number of bits is 80 × 25 × 10 = 20,000 bits per page. For 50 pages, 1,000,000 bits will be transferred.

The rate of data transfer in serial data communication is stated in bps (bits per second); also *baud rate*.

Baud and bps rates are *not* necessarily equal.

17.1: BASICS OF SERIAL COMMUNICATION data transfer rate

Example 17-2

Calculate the time it takes to transfer the entire 50 pages of data in Example 17-1 using a baud rate of:
(a) 9600 (b) 57,600

Solution:

(a) 1,000,000 / 9600 = 104 seconds
(b) 1,000,000 / 57,600 = 17 seconds

Baud rate is modem terminology, defined as number of signal changes per second. As far as the conductor wire is concerned, baud rate and bps *are* the same

17.1: BASICS OF SERIAL COMMUNICATION data transfer rate

Example 17-3

Calculate the time it takes to download a movie of 2 gigabytes using a telephone line. Assume 8 bits, 1 stop bit, no parity, and 57,600 baud rate.

Solution:

$2 \times 1{,}000{,}000{,}000 \times 10 \ / \ 57{,}600 = 347{,}222$ seconds = 4 days

Data transfer rate depends on communication ports incorporated into the system, with today's x86 transferring data at rates as high as 115,200 bps.

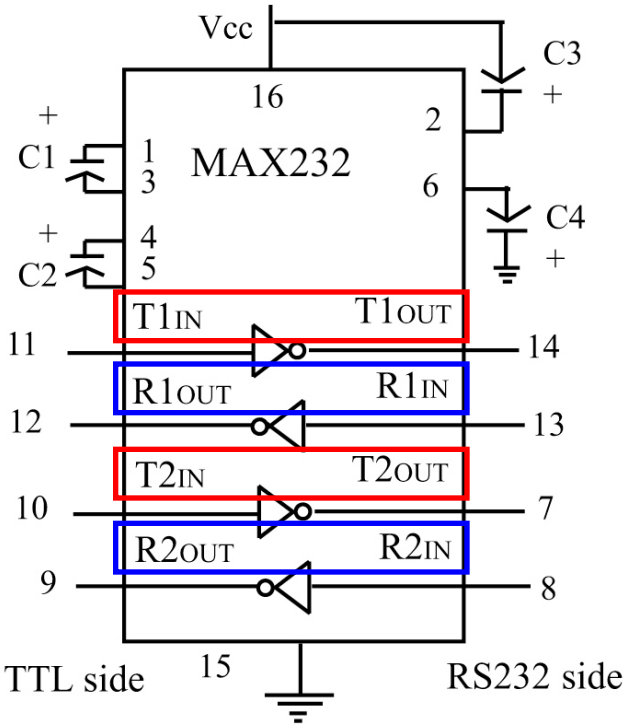
In asynchronous serial data communication, baud rate is generally limited to less than 100,000 bps using a telephone line & modem.

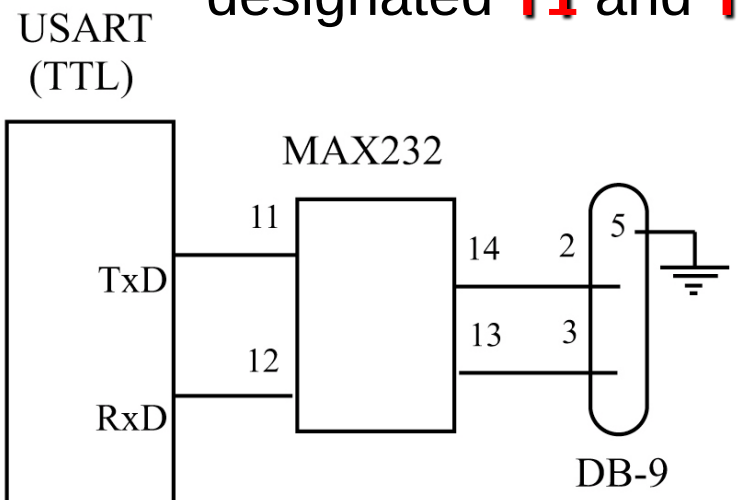
17.1: BASICS OF SERIAL COMMUNICATION
RS232 and other serial I/O standards

- For compatibility in data communication equipment, an interfacing standard called RS232 was set by the Electronics Industries Association (EIA) in 1960.
  - Today, the most widely used serial I/O interface standard.
- RS232 input/output voltage is not TTL compatible:
  - Bit 1 is –3 to –25 V; Bit 0 +3 to +25 V.
  - +3 to –3 is *undefined*.
- To connect RS232 to a TTL-level chip requires voltage converters to convert the TTL logic levels to the RS232 voltage level, and vice versa.
  - MAX232/233 IC chips commonly called *line drivers*.

17.1: BASICS OF SERIAL COMMUNICATION
RS232 and other serial I/O standards

- The MAX232 has two sets of line drivers for transferring and receiving data





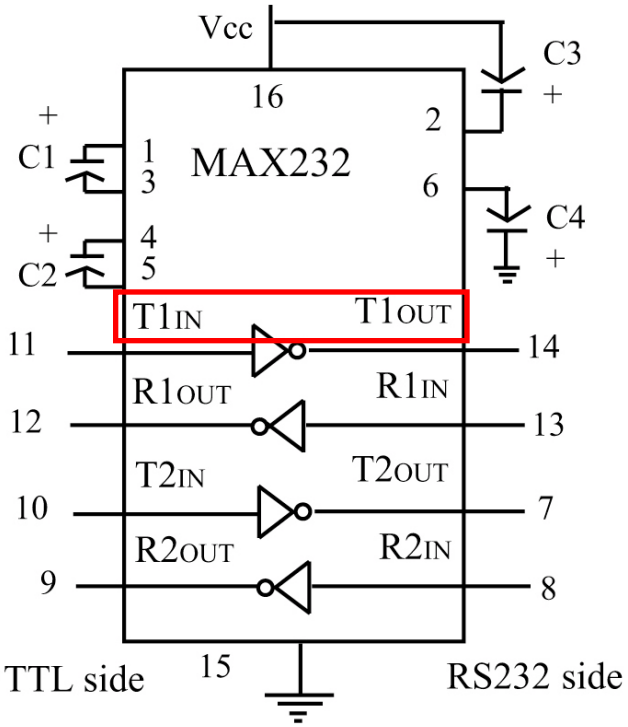
Line drivers used for TxD are designated **T1** and **T2**

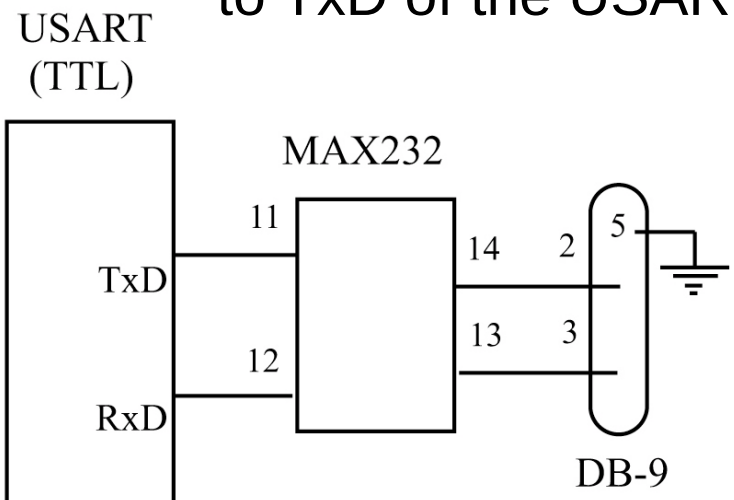
Line drivers used for RxD are designated **R1** and **R2**.

Fig. 17-4 Inside MAX232 and its connection to the USART

17.1: BASICS OF SERIAL COMMUNICATION
RS232 and other serial I/O standards

- The MAX232 T1 line driver has a designation of T1in & T1out on pin numbers 11 & 14, respectively.





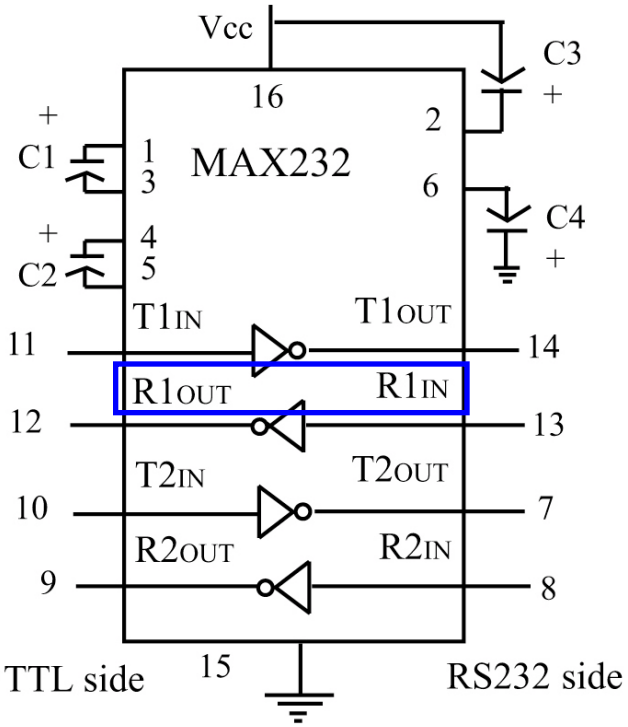
T1in pin is the TTL side, connected to TxD of the USART.

T1out is connected to RxD of the RS232 DB connector.

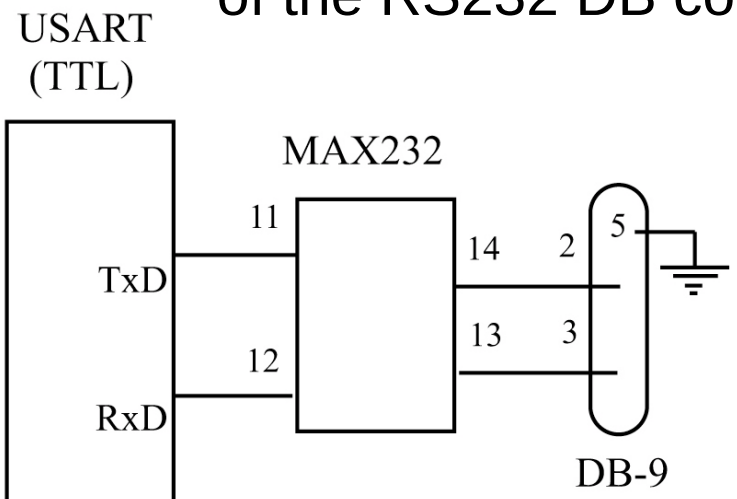
Fig. 17-4 Inside MAX232 and its connection to the USART

17.1: BASICS OF SERIAL COMMUNICATION
RS232 and other serial I/O standards

- The R1 line driver has a designation of R1in and R1out on pin numbers 13 and 12, respectively.



R1in (pin 13) is connected to TxD of the RS232 DB connector.

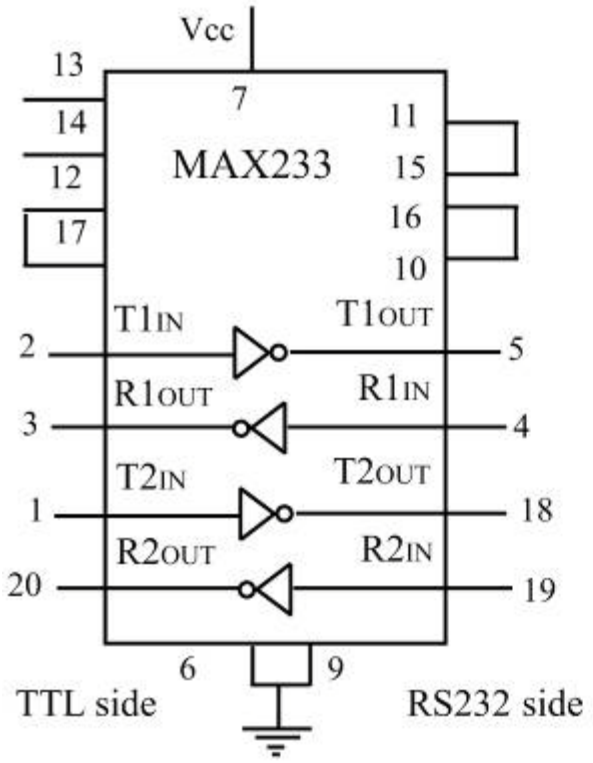


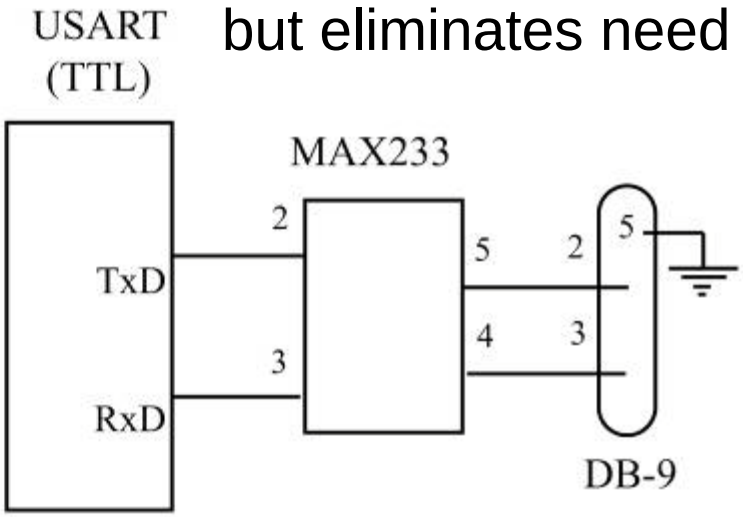
R1out (pin 12) is the TTL side, connected to RxD of the USART.

Fig. 17-4 Inside MAX232 and its connection to the USART

17.1: BASICS OF SERIAL COMMUNICATION
RS232 and other serial I/O standards

- MAX232 requires four capacitors ranging from 1 to 22 mF—some designers use the MAX233





Performs the same as MAX232, but eliminates need for capacitors.

MAX233 is a much more expensive chip than MAX232.

Fig. 17-5 Inside MAX233 and its connection to the USART

17.1: BASICS OF SERIAL COMMUNICATION
RS232 pins

Pins and their labels for the RS232 cable, which is commonly referred to as the DB-9 connector.

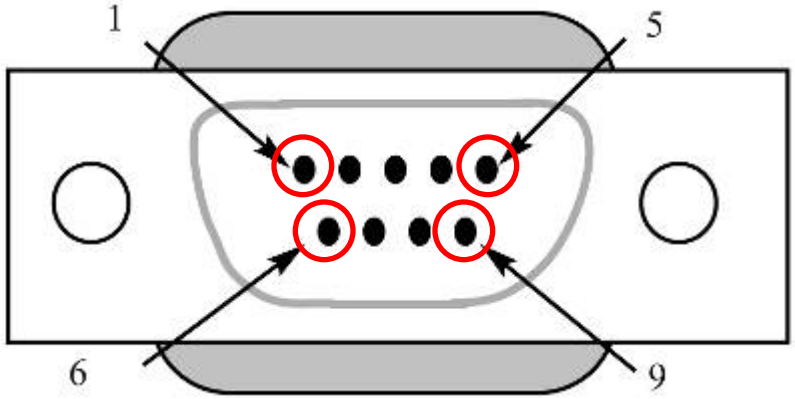


Fig. 17-6 DB9 9-Pin Connector

Table 17-1: IBM PC DB-9 Signals

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)

17.1: BASICS OF SERIAL COMMUNICATION
data communication classification

- **DTE** (data terminal equipment) refers to terminals and computers that send and receive data.
- **DCE** (data communication equipment) refers to communication equipment, such as modems, responsible for transferring the data.

17.1: BASICS OF SERIAL COMMUNICATION
data communication classification

- The simplest connection between two PCs (DTE & DTE) requires pins TxD, RxD, and ground, also referred to as SG (signal ground). .
  - Connection between two DTE devices, such as two PCs, requires pins 2 and 3 to be interchanged

RS232 signal definitions are from the point of view of DTE.

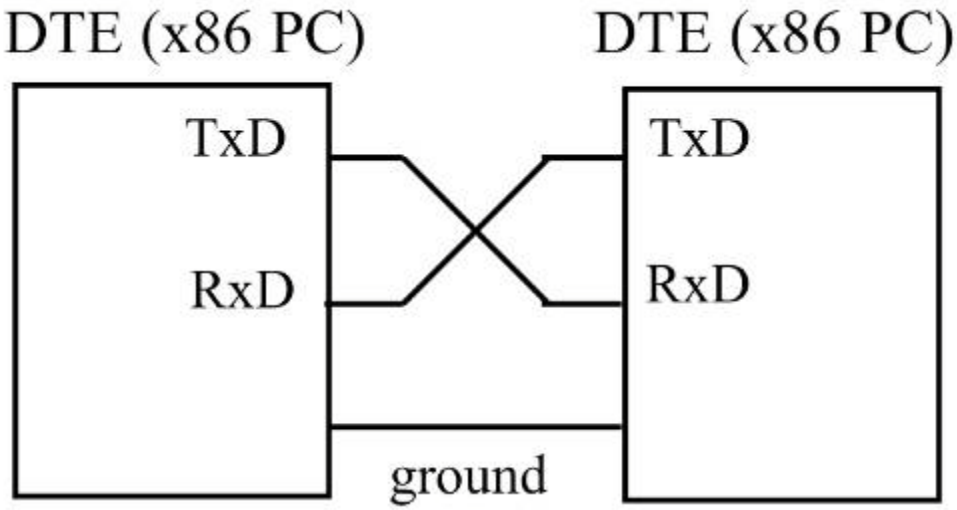


Fig.17-7 Null Modem Connection for Data Lines

17.1: BASICS OF SERIAL COMMUNICATION
examining the RS232 handshaking signals

- To ensure fast & reliable data transmission between two devices, the data transfer must be coordinated.
  - Some RS232 pins are used for handshaking signals.
- **DTR** (data terminal ready) - when a terminal or PC COM port is turned on, after self-test, it sends signal DTR to indicate it is ready for communication.
  - An output from DTE (COM port); an input to the modem.
- **DSR** (data set ready) - when a DCE (modem) is turned on and has gone through self-test, it asserts DSR to indicate that it is ready to communicate.
  - An *active-low* signal.

17.1: BASICS OF SERIAL COMMUNICATION
examining the RS232 handshaking signals

- **RTS** (request to send) - when the DTE has a byte to transmit, it asserts RTS to signal the modem that it has a byte of data to transmit.
  - An *active-low* output from DTE; an input to the modem.
- **CTS** (clear to send) - when the modem has room for storing the data it is to receive, it sends signal CTS to DTE (PC) to indicate it can receive the data.
- **CD** (carrier detect, or DCD, data carrier detect) - the modem asserts signal CDC to inform DTE (PC) that a valid carrier has been detected and that contact between it and the other modem is established.

17.1: BASICS OF SERIAL COMMUNICATION
examining the RS232 handshaking signals

- **RI** (ring indicator) - an output from modem (DCE) and input to a PC (DTE), that indicates that the telephone is ringing.
  - ON and OFF in synchronization with the ringing sound.

17.1: BASICS OF SERIAL COMMUNICATION
examining the RS232 handshaking signals

- RTS and CTS actually control the flow of data.
  - When the PC wants to send data it asserts RTS
    - If the modem is ready to accept the data, it sends CTS.

RTS and CTS are also referred to as *hardware flow control* signals.

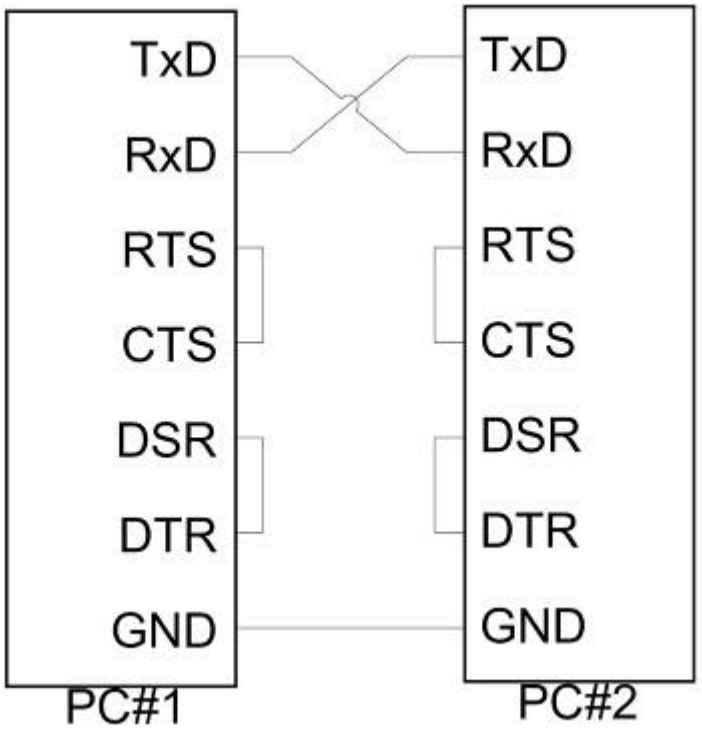


Fig.17-8
Null Modem Connection with Control Signals

17.2: PROGRAMMING x86 PC COM PORTS
IBM PC COM ports

- In the x86 PC, as many as four COM ports can be installed, numbered 1, 2, 3, and 4.
  - BIOS numbers them as 0, 1, 2, and 3.
- The PC POST (power-on self-test) tests the USART chip for each of the four COM ports.
  - If they are installed, their I/O port addresses are written to memory locations 0040:0000–0040:0007.

17.2: PROGRAMMING x86 PC COM PORTS using HyperTerminal on x86 PC

- HyperTerminal is a widely used utility that comes with Windows 2000/XP, for communication with the x86 PC via the serial COM port.
  - It is not dynamic, so it cannot be incorporated into programs

Table 17-2: Some HyperTerminal Baud Rates

Baud Rates
300
600
1,200
2,400
4,800
9,600
19,200
38,400
57,600
115,200

17.2: PROGRAMMING x86 PC COM PORTS
programming COM ports using BIOS INT 14H

- Serial communication ports of the x86 PC can be accessed using the BIOS-based INT 14H.
  - Various options of INT 14H are chosen with the AH registers.

See the entire list of BIOS INT14 functions in Fig. 17-9 on pages 456 - 457.

AH INT 14H Function

00 Initialize COM Port

Additional Call Registers
AL = parameter (see below)
DX = port number (0 if COM1, 1 if COM2, etc.)

Result Registers
AH = port status (see below)
AL = modem status (see below)

Note 1: The parameter byte in AL is defined as follows:

7 6 5 4 3 2 1 0	Indicates
x x x	Baud rate (000 = 110, 001 = 150, 010 = 300, 011 = 600, 100 = 1200, 101 = 2400, 110 = 4800, 111 = 9600)
x x	add, 11 = even, x0 = none)

17.2: PROGRAMMING x86 PC COM PORTS
programming COM ports using BIOS INT 14H

- Using BIOS INT 14H we can send and receive characters with another PC via a COM port:
  - 1. To send a character…
    - INT 14H, AH = 1, AL = character.
  - 2. To receive a character…
    - INT 14H, AH = 3 (to get COM port status in register AH)
  - 3. To read the received character…
    - INT 14H, AH = 2 (AL holds the character upon return)

17.2: PROGRAMMING x86 PC COM PORTS
programming x86 COM port using C# 2005

- To program the COM port we can use System.IO.Ports namespace in C# 2005.
  - Shown in Programs 17-2 and 17-3 on page 459.
    - Program 17-2 writes data to the COM port.
    - Program 17-3 reads data from the COM port.

The x86 PC

assembly language, design, and interfacing

fifth edition

Prentice Hall

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY