Serial Interfacing

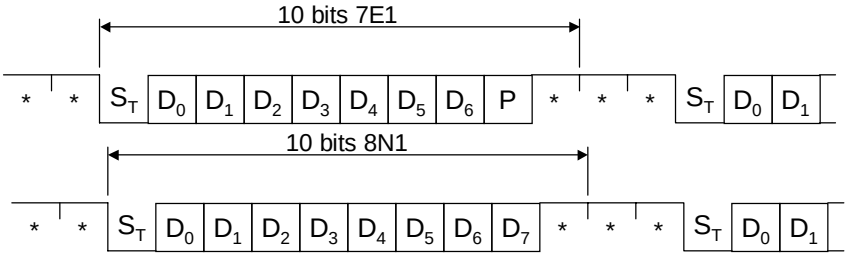
Serial Data Transfer – used by keyboards, plotters, modems and other peripherals with low data transfer rates (low bandwidth)

2 Types:

Asynchronous – CPU and device are not using a common time reference
- no common clock/timing signal
- special bit patterns indicate beg/end
- slower than synchronous

Synchronous – common time reference used
- timing controlled
- synchronization pulses related to a clock are sent/received
- faster than asynchronous

Asynchronous Frame



Mark – A Constant Logic-1 Denoted by * *Space* – A Constant Logic-0

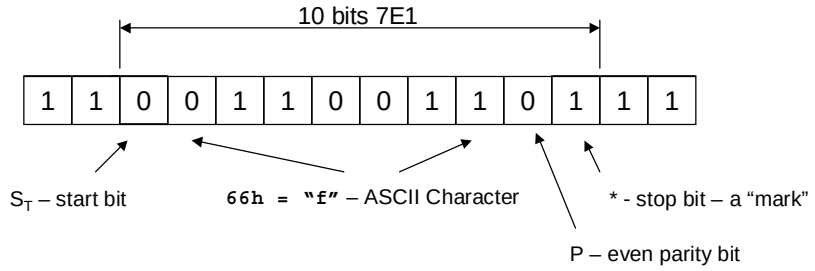
Standard is for Serial Line to CONSTANTLY be Driven to a MARK While Inactive

S_T – start bit
D_0 – LSB
D_6(D_7) – MSB
P – parity bit
* - stop bit – a "mark"

1 ASCII char. = 7 bits
(e.g. `DEL = 7fh` – higher is PC specific)
Typical is 10 bits for asynchronous transfer
Serial data with even/odd parity

Serial Data Receiver Starts Processing When:
1) high to low is sensed (start bit detection)
2) following (7 or 8) bits represent a character
3) parity bit for error detection
4) stop bit is detected (a "mark")

Frame Example

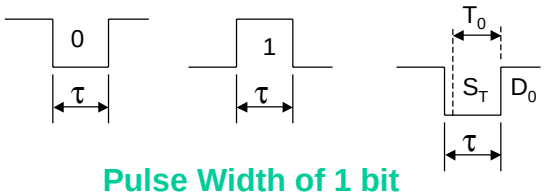


When Receiver "sees" a Start Bit (high to low transition):

1) Local Timer Starts
2) Each bit sampled at midpoint in time (± % clock tolerance)
3) Maximum tolerance is ± ½ of 1 bit time interval over 10 intervals

= (½)/10 = **5%**

Electrical Parameters *(asynchronous)*

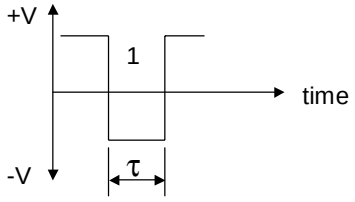


Pulse Width of 1 bit

T_0 – Start Time for Receiver Clock

Why is T_0 less than τ ?

***ANSWER*:** Circuit Delay for Falling Edge Detection Circuit



- Voltage Levels for 0, 1
- Typically ***NOT*** TTL Levels
- Negative Logic Normally

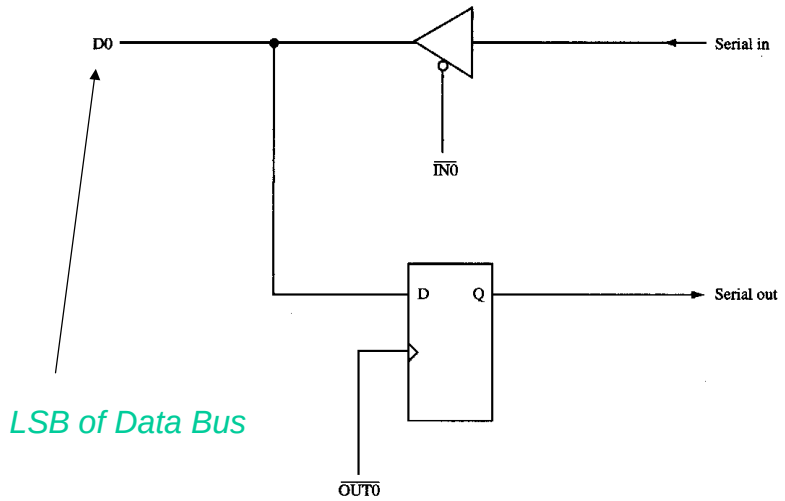
Baseband Signaling Speeds

Baud Rate	Bits per second (bps)	Real Period (s)	compression
75	75	13.3 ms	NONE
110	110	9.1 ms	NONE
150	150	6.7 ms	NONE
300	300	3.3 ms	NONE
600	600	1.67 ms	NONE
1200	1200	833µs	NONE
2400	2400	417µs	NONE
4800	4800	208µs	NONE
9600	9600	104µs	NONE
9600	14400	104µs	YES
19200	19200	52µs	YES
19200	28800	52µs	YES
19200	36600	52µs	YES
19200	56000	52µs	YES

Baud Rate vs Bits Per Second

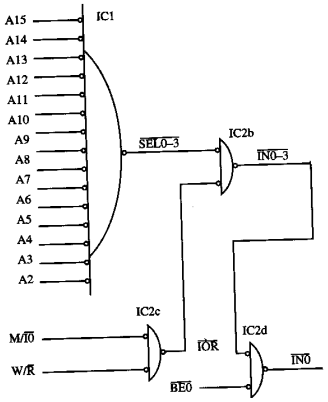
- Baud rate is the rate at which signaling events are sent
- Bits per second (bps) is the number of bits transferred per second (any type of bits, data or overhead bits)
- If only a '1' or '0' is sent for each signaling event, then baud rate = bps
- However, could use a signaling protocol that transfers multiple bits per signaling event
  - i.e., use 4 different voltage levels, send two bits of data per signaling event (00 = -15v, 01= -5v, 10=+5v, 11 = 5v).
  - In this case, bit rate will be double the baud rate
- The effective data rate is the rate at which data is transferred, minus the overhead bits (ie. start and stop bits).

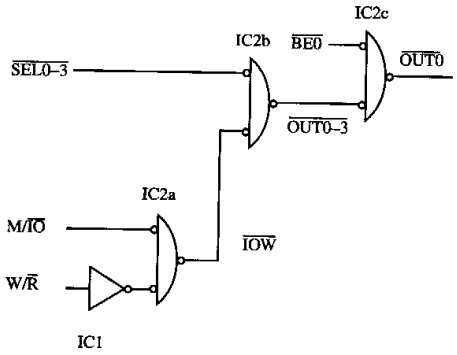
Serial Port is 1-bit Parallel Port




```
out   0,   al
in    al,  0
```

Serial Port Enable Signals






```
in    al,  0
```

```
out   0,   al
```

Recovering/Generating Serial Data

- Timing Loop-

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                  ;
;  Serial port timing loop for transmit operation for PC without  ;
;  UART.  Upon entry the 8 bit word to be transmitted must be in  ;
;  al.  This procedure destroys the content of al, CF and cx.     ;
;                                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            EXTERN      DELAY:NEAR
            DPORT       EQU   00H
      CODE  SEGMENT
            ASSUME      CS:CODE

FIG10_3     PROC        NEAR
            mov   cx,         10          ;Initialize the bit count
            clc                           ;CF=0, CF is the start bit
            rcl   al,         1           ;Rotate CF into bit 0 of al
TRANS:      out   DPORT,      al          ;I/O write to 00H of al (0)
            call  DELAY                   ;Wait for I/O write to finish
            rcr   al,         1           ;Rot data into bit 0 of al
            stc                           ;CF=1 (stop bit)
            loop  TRANS                   ;All 10 bits xmitted?
            ret                           ;10 bit transmission complete
FIG10_3     ENDP
            CODE  ENDS
            END
```

AL/CF Content During Timing Loop

(assume A5h is to be serially transmitted)

CF	AL	
X	A5h	;proc entry
0	A5h	;clc
1	4Ah	;rcl
0	A5h	;rcr
1	A5h	;stc
1	D2h	;rcr
1	D2h	;stc
0	E9h	;rcr
1	E9h	;stc

CF	AL	
1	F4h	;rcr
1	F4h	;stc
0	FAh	;rcr
1	FAh	;stc
0	FDh	;rcr
1	FDh	;stc
1	FEh	;rcr
1	FEh	;stc
0	FFh	;rcr
1	FFh	;stc

AL/CF Content During Timing Loop

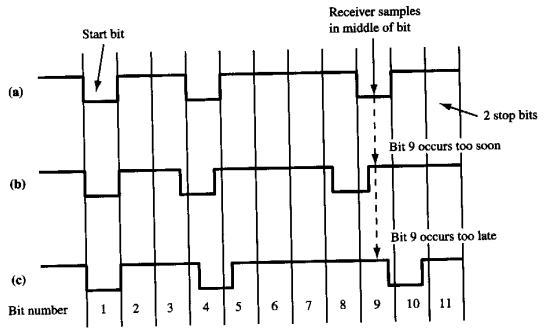
(CONTINUED)

CF	AL	
1	FFh	;rcr
1	FFh	;stc
1	FFh	;rcr
1	FFh	;stc
1	FFh	;return from procedure

Synchronization of Serial Data

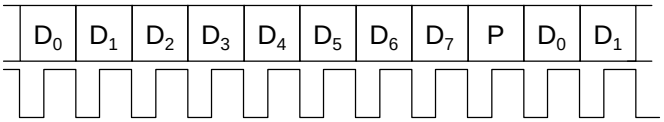
- R_x and T_x Clocks may Differ Slightly in Period
- Sampling is Done at Middle of Bit Time Interval
- Can Result in Accumulation of Error over 10-bit Time Interval
- Timing is Reset at Beginning of Frame (Start Bit)
  - Self-Synchronizing

Figure 10.5 (a) Serial data transmitted at the proper rate. (b) The data rate is too fast. (c) The data rate is too slow.



Synchronous Serial Data

1) No start/stop bits
2) Data (and parity *maybe*) only



Advantage: Faster (since no start/stop bits)
Disadvantage: Clock must be transmitted with data

Initially a sync-frame is sent followed by a block of data (which can be many characters)

Universal Asynchronous Receiver/Transmitter

UART

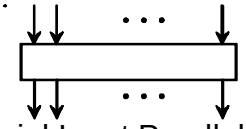
Special Circuit that Relieves Processor from Executing Timing Loops

UART Characteristics

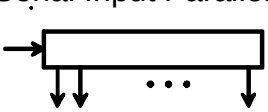
- *Appears as a Parallel I/O Port to the x86*
- *Contains Both Rx and Tx Circuits*
- *Contains Status Registers (BUSY/READY and ERROR Conditions)*
- *Types of Errors UART Can Detect*
  - Framing Error
  - Parity Error
  - Overrrun
  - Invalid Start Bit Received
  - Single Bit Data Error Detected
  - Stop Bit Not Found
- *Most UARTs use a Dedicated 16x Clock Signal*
  - 1200 bps $\rightarrow$ 19.2 kHz Clock Signal
  - Each Bit Divided into 16 “Time Slices”
- *Rx Circuit Converts Serial to Parallel – Tx Converts Parallel to Serial*
  - Combination PISO/SIPO Registers

Register Types

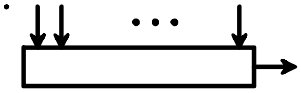
- PIPO - Parallel Input Parallel Output



- SIPO - Serial Input Parallel Output



- PISO - Parallel Input Serial Output



- SISO - Serial Input Serial Output

PC16550D UART

Transmit Functionality

1) *Receives 1 Byte from Processor*
2) *Converts to Serial Form (PISO Register)*
3) *Adds Start, Stop and Parity bits*
4) *Clocks Data Out Serially (Possible Rates are 0-256 kbps)*

Receive Functionality

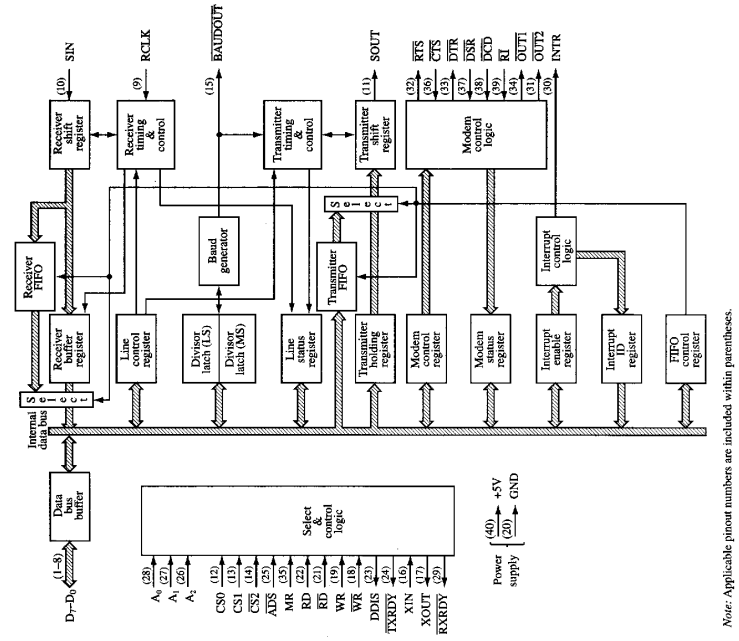
1) *Receives Serial Frame from Device*
2) *Converts to Parallel Form (SIPO Register)*
3) *Checks for Errors (Framing, Parity, Overrun)*
4) *Stores Received Byte for Processor Access*

Supported I/O Control Schemes

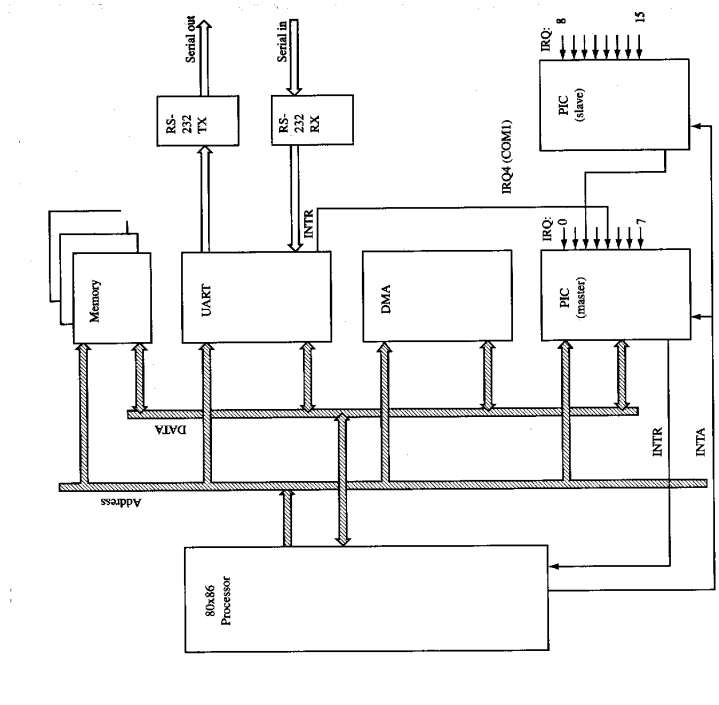
1) *Polling (Parallel)*
2) *Interrupts*
3) *DMA*

Successor of the NS8250/8251 and 16540
Need 1 16550 per Serial Port (typically 2 per PC since COM1 and COM2)
NPC16552D is Single Package Device Containing Equivalent of 2 16550s

16550 Block Diagram

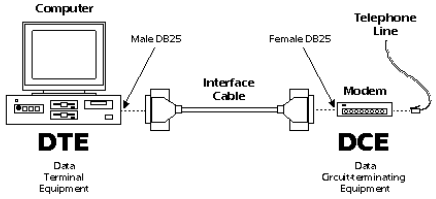


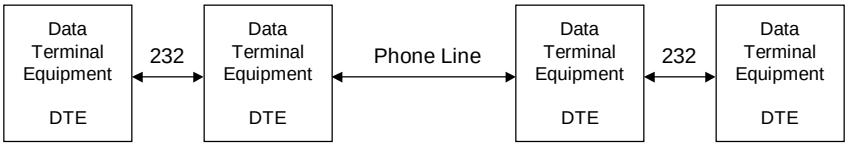
PC Interface



EIA RS-232 Serial Interface Standard

- EIA – Electronic Industry Associates
- First RS-232 Standard in 1969 for:
  - CPU to Terminal
  - Terminal to Terminal
  - CPU to CPU
  - Initially Assumed Phone Lines used with Modems at Each End
- Latest is EIA232E in 1991 (no longer RS, but everyone still calls it that)



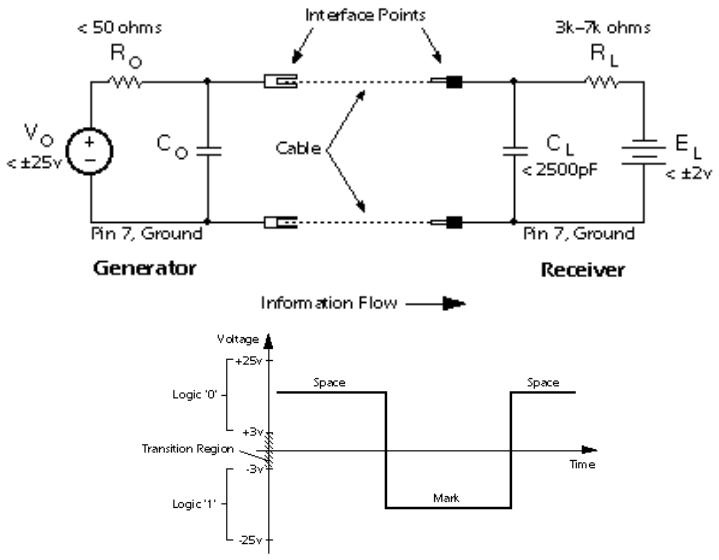


Basic 232 Specifications

- Maximum Data Rate = 19,200 bps
- Maximum Cable Length = 50 feet
- Maximum Capacitance/foot = 50 pF
- |Vmax| = ±25V
- Negative Logic:
  logic-1 = -12V (typ) logic-0 = +12V (typ)
- 2 Standard Connectors and Pinouts
  - DB9 is the 9-pin Connector
  - DB25 is the 25-pin Connector

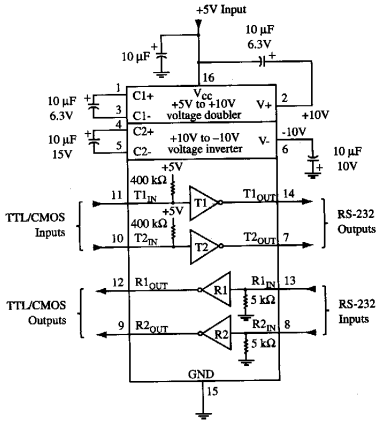
**NOTE: DB25 is a General Connector NOT an RS-232 Only; PC Parallel Ports Commonly Use the DB25 Also!*

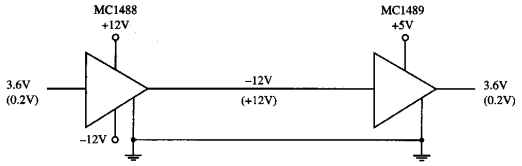
Basic 232 Specifications



TTL Interfacing

- Need “Line Driver” Circuits
- These Convert from TTL Logic Level Voltages to RS-232 Levels

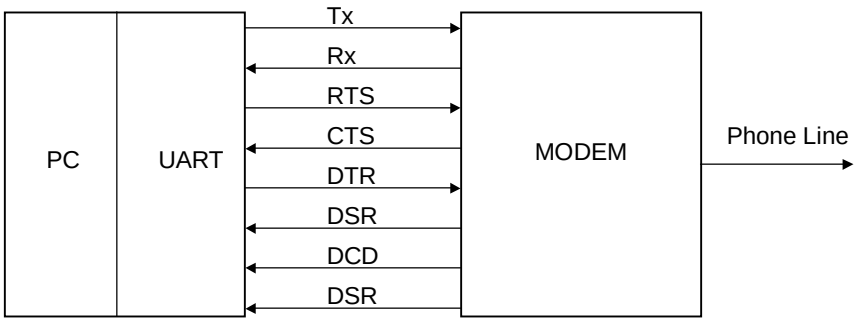




Control Signals

names defined from DTE point of view!

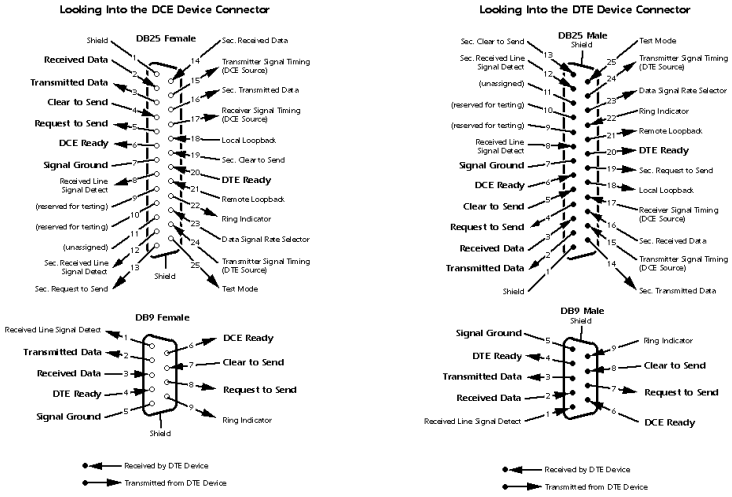
Rx	- Data DTE Receives	
Tx	- Data DTE Transmits	
RTS	- Request to Send Data	(DTE wants to Transmit)
CTS	- Reply	(DCE indicates it is OK to Receive)
DTR	- Data Terminal Ready	(DTE device is present and ready)
DSR	- Data Set Ready	(DCE device is present and ready)
DCD	- Data Carrier Detect	
RI	- Ring Indicator	
GND	- Common for All Above Signals	



All 232 Signals

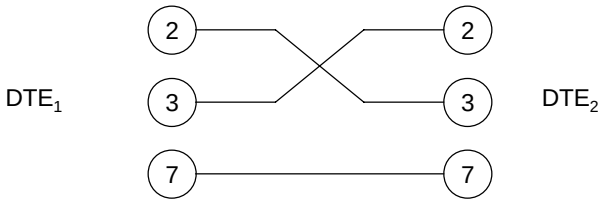
		Data		Control	
Pin	Signal name	From DTE to DCE	To DTE from DCE	From DTE to DCE	To DTE from DCE
1	Protective ground				
2	Transmitted data	X			
3	Received data		X		
4	Request to send ($\overline{RTS}$)			X	
5	Clear to send ($\overline{CTS}$)				X
6	Data set ready ($\overline{DSR}$)				X
7	Signal ground				
8	Data carrier detect ($\overline{DCD}$)				X
9/10	Reserved for data set testing				
11	Unassigned				
12	Secondary data carrier detect				X
13	Secondary clear to send				X
14	Secondary transmitted data	X			
15	Transmit signal element timing				X
16	Secondary received data		X		
17	Receive signal element timing				X
18	Unassigned				
19	Secondary request to send			X	
20	Data terminal ready ($\overline{DTR}$)			X	
21	Signal-quality detector (indicates probability of error)				X
22	Ring indicator				X
23	Data signal rate select (allows selection of two different baud rates)				X
24	Transmit signal element timing			X	
25	Unassigned				

Standard Connector Pinouts



Interfacing Non-Modem Peripherals

- Can Interface Two DTEs
- Must “Cross” Tx and Dx Signals (a Null Modem)



- Syncronization Accomplished by the Asynchronous Frame Bits
  i.e. the Start and Stop bits

Handshaking (2 Main Types)

HARDWARE HANDSHAKING

- uses DTR Signal as a BUSY/READY Indicator

SOFTWARE HANDSHAKING

- Uses ASCII Characters in Data Transmission
- Typically a “Request Transmission Halt” is a
  XOFF (ASCII control Char **13h**)
  OR **CTRL-S**
- Typically a “Request Transmission Resume” is a
  XON (ASCII control Char **11h**)
  OR **CTRL-Q**