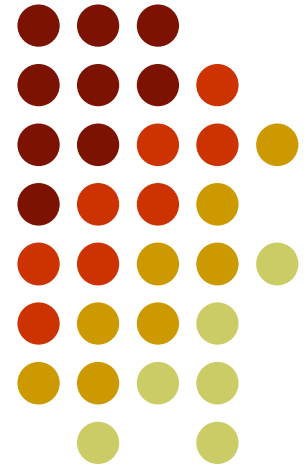
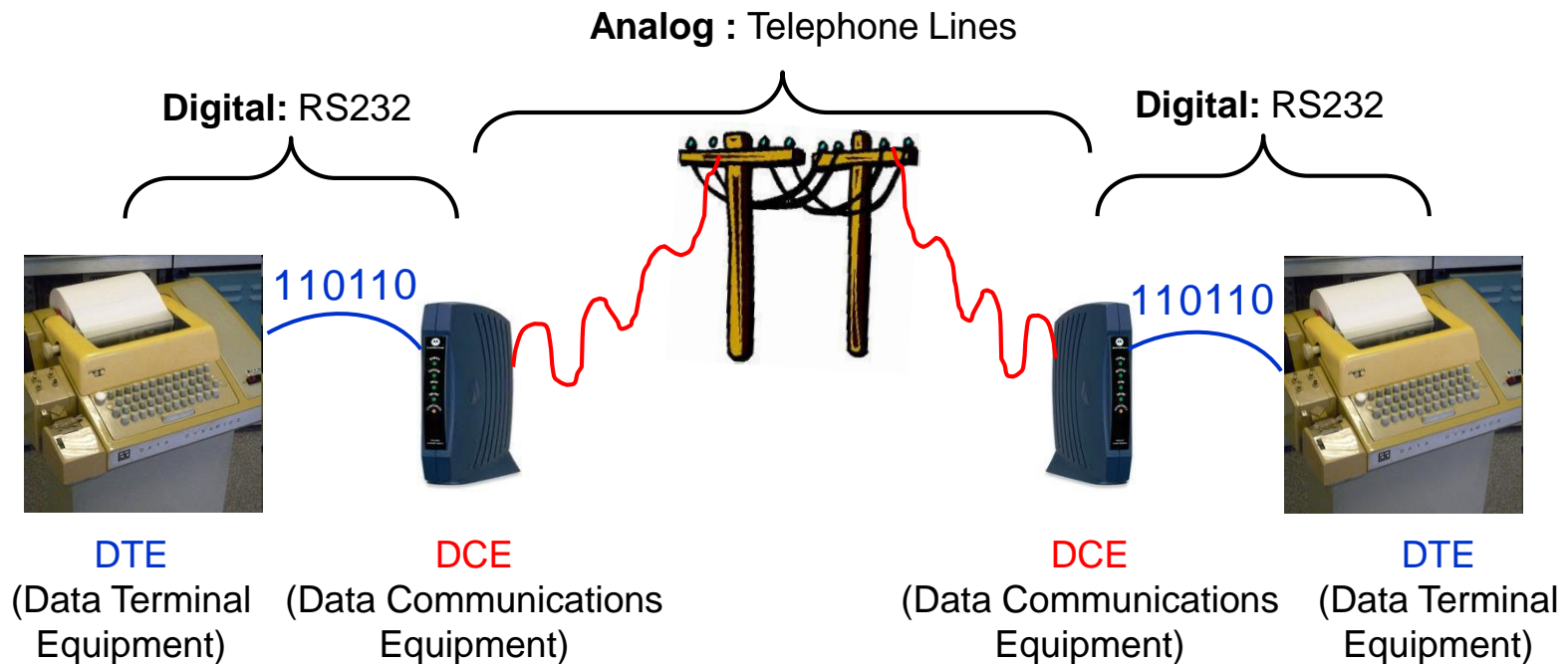


# Asynchronous Serial Communications

- **Origins of Serial Communications**
  - Modems and Telephones, DTE/DCE equipment
- **Digital Serial Communications**
  - Simplex, half-duplex and full-duplex
  - Asynchronous serial transmission
  - The Motorola 6850 (ACIA)
- **The RS-232C Standard**
  - Mechanical and electrical specs
  - Control lines
  - Handshaking
  - Null modem



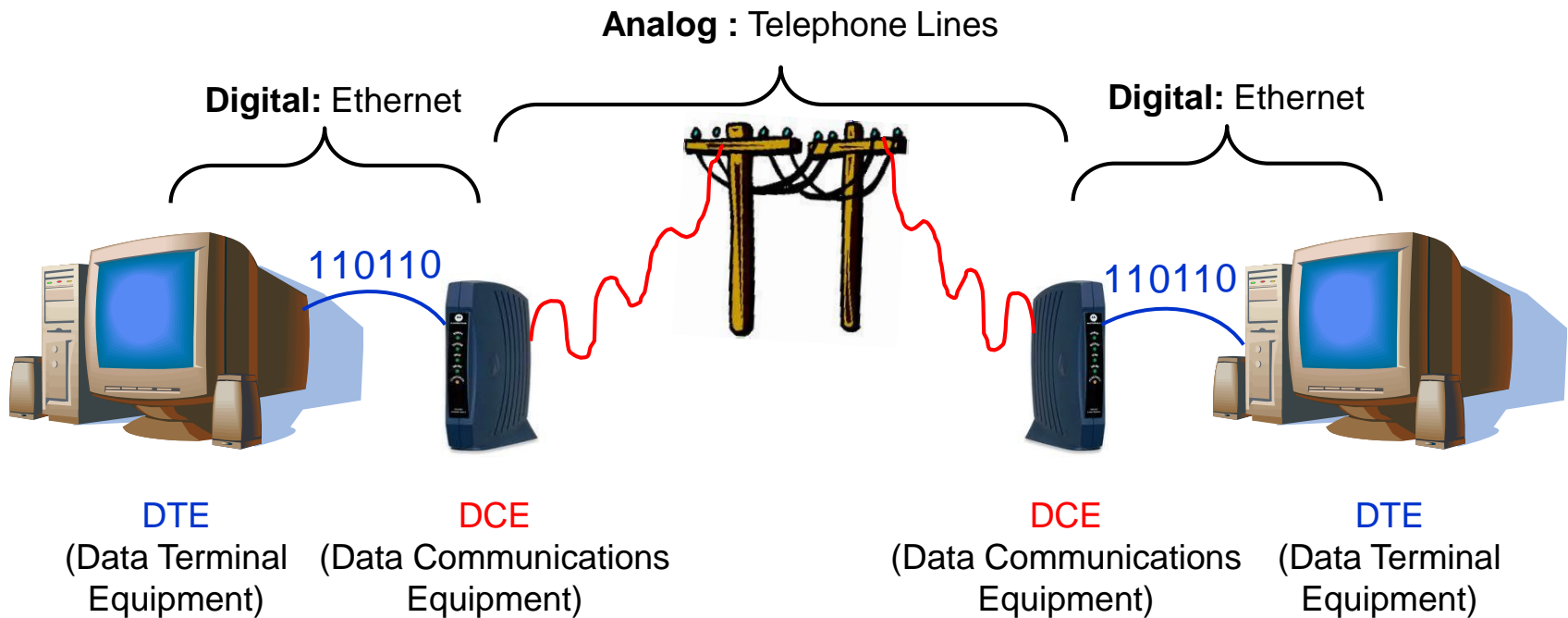
# Origins of Serial Communications and RS232



## Communications over the Telephone Line

- **DTE** equipment lies at each end. It is the origin and destination of data. Typical DTE equipment today includes **computers, printers terminals** etc.
- **DCE** equipment sits between **DTE** equipment and converts digital data into whatever form is required to transmit that data over the communications medium, e.g. telephone lines, fibre optic, wireless comms etc. A typical piece of **DCE** equipment is a **MODEM** or Modulator/demodulator
- **RS232** is a digital communications standard to link **DTE** to **DCE** equipment 2

# Serial Communications and the Internet



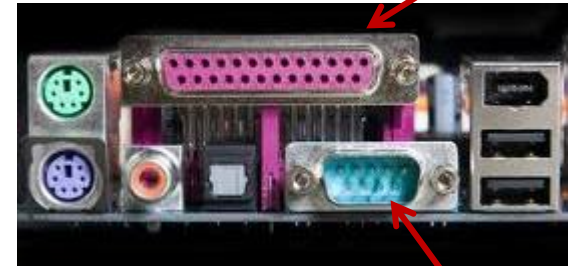
- We use those origins today as the basis of the internet, with high speed modems linking computers around the world, however, the **RS232** link has been superseded by **Ethernet** for massively increased speed.
- **RS232** has also been replaced by other forms of serial communications such as **USB**, **Serial ATA** (disk drives) and **Firewire**. Although faster, these are hugely more complex than **RS232** from both a S/W and H/W perspective.
- **RS232** is still the choice for low speed easy to use communications.

# Serial Communications and the Internet

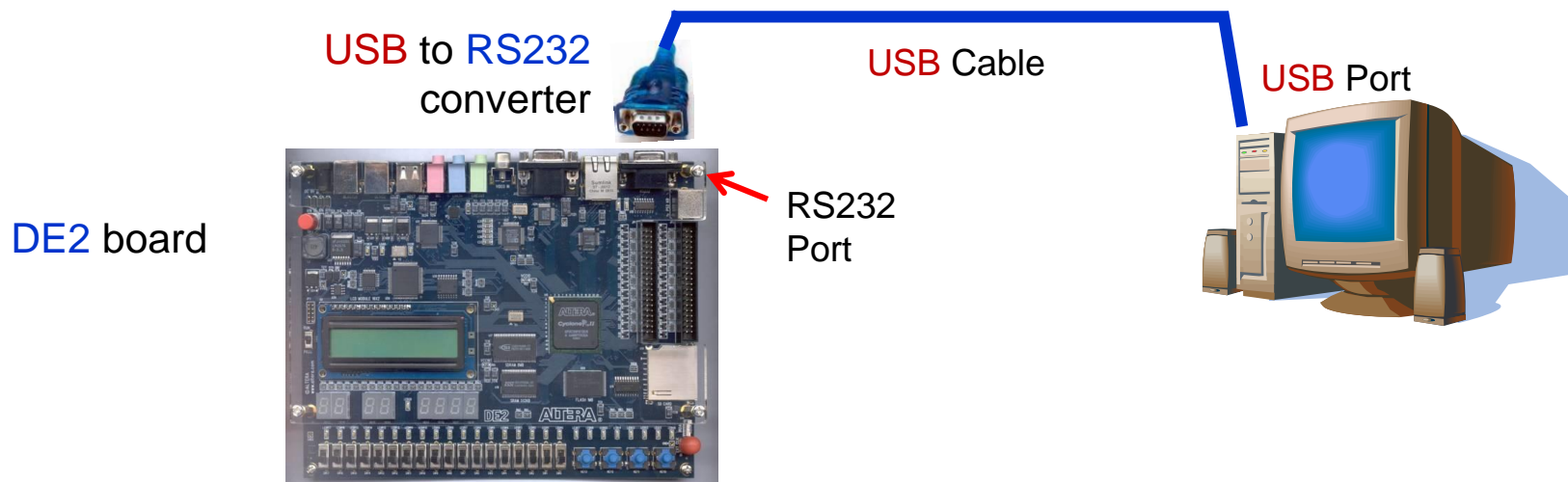


- A few years ago, PCs were equipped with **RS232** and **Printer** ports, but the size of the connectors and the limited applications for them (*in the PC world*) means that they are no longer fitted as standard.
- However many small embedded computers and peripherals still have **RS232** ports. You can use a **USB/RS232** converter to allow **RS232** communications via the PC's USB ports.

Parallel Printer port on early PC motherboard



RS232 Serial port on early PC motherboard

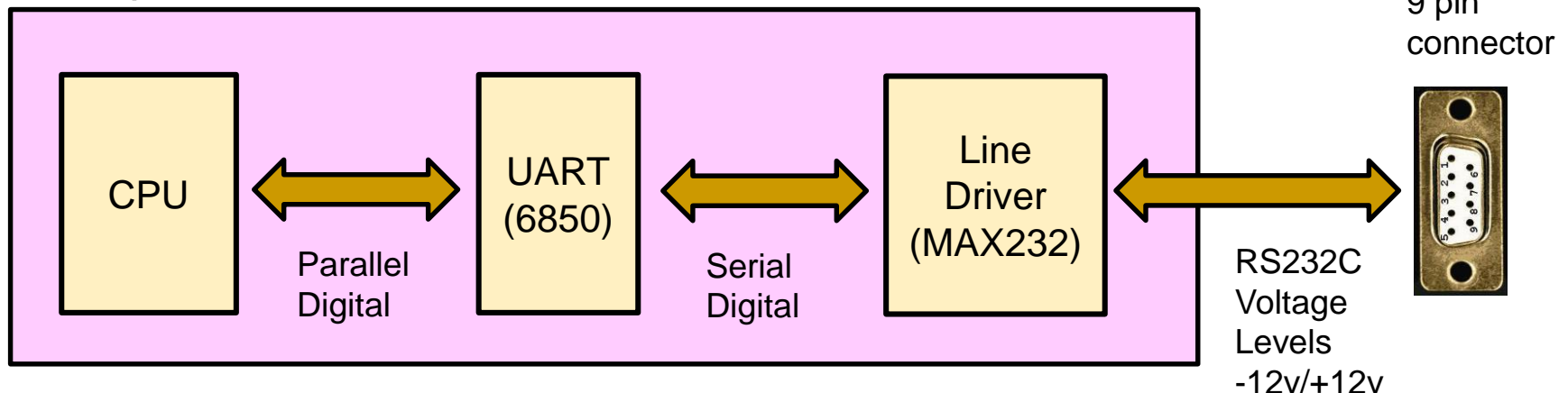


# Asynchronous Serial Communications



- **RS-232** is a standard defined by the Electronic Industries Association ("EIA").
- There are three flavours of RS-232 (**A**, **B**, and **C**) with each one defining a different voltage range. The most commonly used variety is **RS-232C**, which uses a voltage range of between **-12V** to **-3V** to represent **logic 1**'s and **+3v** to **+12v** to represent **logic 0**'s in the digital data stream.
- **RS-232C** cable lengths are limited to about 25 feet (8m).
- Computers use some kind of UART (Universal Asynchronous Receiver/Transmitter) to convert between parallel (byte wide) data and serial data e.g. the 6850 chip.
- "Line driver" chips then convert the stream of digital 0s and 1's coming from the UART into the voltage levels required by the RS232 standard before they emerge from a suitable connector.
- **NOTE:** If you don't want RS232 voltage levels you can just use the digital data stream directly from the UART this allows for a simple 3 wire 5 volt serial communications

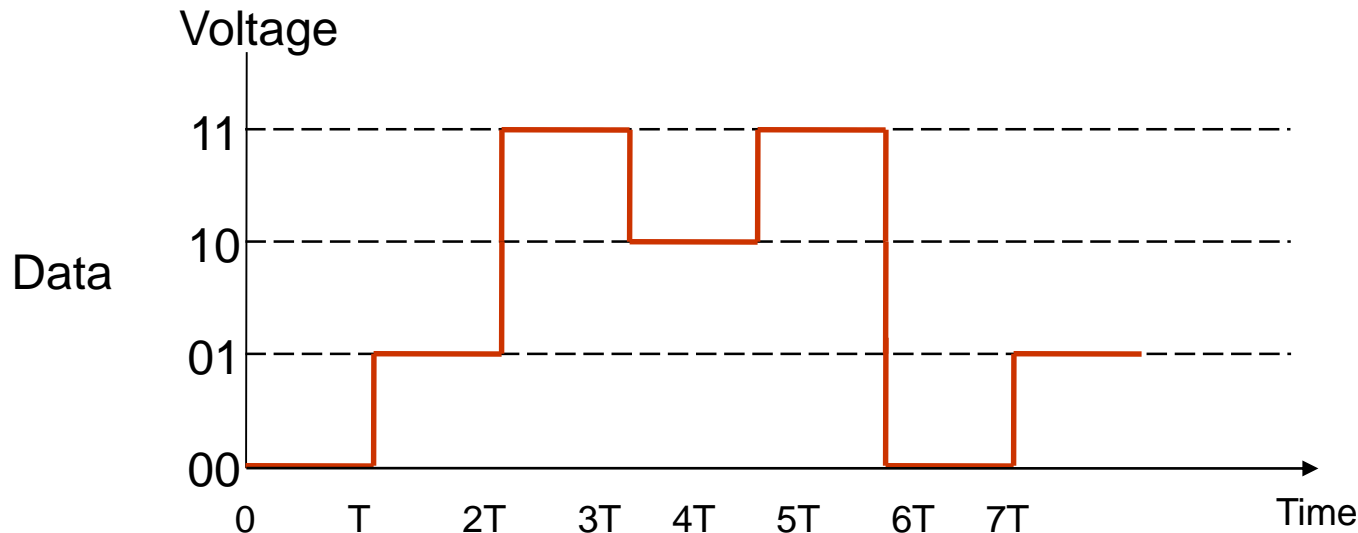
## Computer



# Some Terminology used in Serial Transmission



- **Baud rate** - is a measure of the *switching speed* of a signal, that is, the number of times per second that a signal changes *state* or *transitions*.
- **Bits per second (bps)** – is a measure of the rate at which data can be transmitted. For example, a slow dial up modem may quote only 56K bps
- **Baud rate** and **bps** do not always mean the same thing. A signal may transition ‘ten’ times per second i.e. its baud rate is ten, but if the signal’s amplitude has 4 distinct voltage levels then each transition can convey 2 bits of information, i.e. its bps is **20**.



- For digital serial comms: **BPS** = **Baud Rate** since the signal only transitions between two voltage levels. RS232 always refers to “**baud rate**” as a measure of **BPS**.
- Industry Standard baud rates are: **110, 300, 600, 1200, 4800, 9600, 14400, 19200, 38400**, and so on.

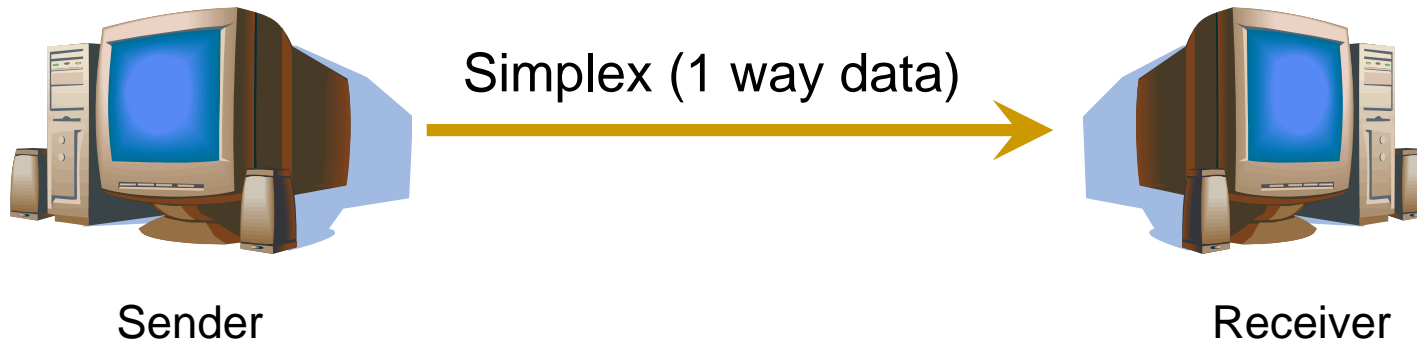
# Modes of Communications



- Regardless of how the communication is implemented (e.g. serial, parallel, cable, fibre optic etc.), when two pieces of equipment are connected together, we have a choice about the way in which they are able to communicate.

## Simplex Communication (Uni-directional)

- With a simplex connection, data can only flow in **one** direction, in much the same way that traffic is allowed to travel down a one-way street.
- An example of simplex communication is the **television** or **radio**.
- Simplex channels are rarely used in computer based systems because it is not possible for the receiver to send back any **error**, **control** or **acknowledge** signals to the sender.

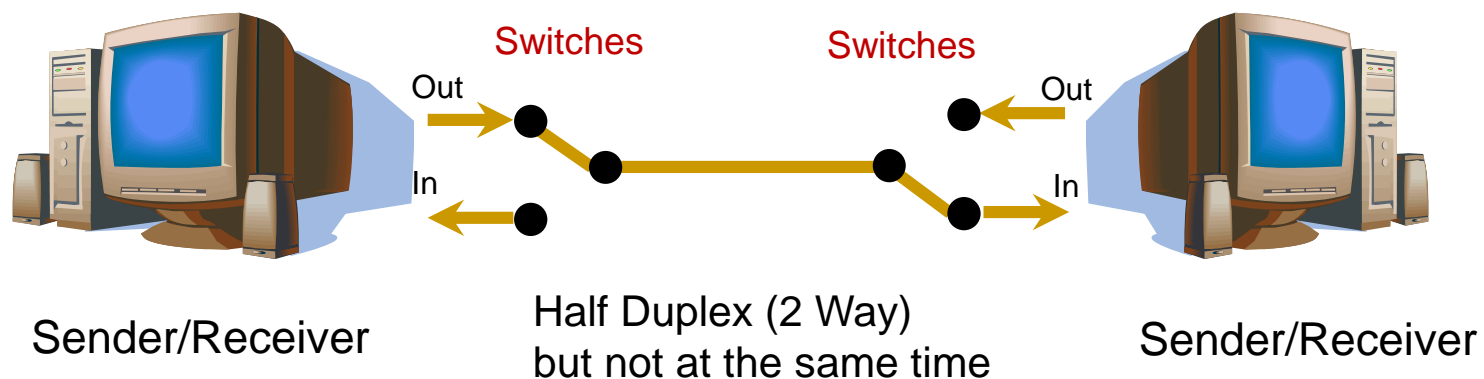


# Modes of Communications



## Half Duplex Communications (Bi-directional)

- A half duplex connection can send and receive, but not at the same time. It's like a one-lane bridge where two way traffic must give way in order to cross.
- For bi-directional communications, each end has to reconfigure itself for transmitting or receiving. Computer controlled electronic switches are employed within each piece of equipment to dynamically switch data direction.
- As with the bridge, a problem of synchronisation always exists, i.e. who speaks first. Usually there is a master/slave relationship. The master (e.g. computer) controls all communication and the slave only speaks when spoken to (e.g. disk drive).
- An example of a Half-duplex communication system might be a [walkie-talkie](#), with a separate speak/listen button. [USB](#) and [ethernet](#) uses half duplex communications because it results in fewer wires.



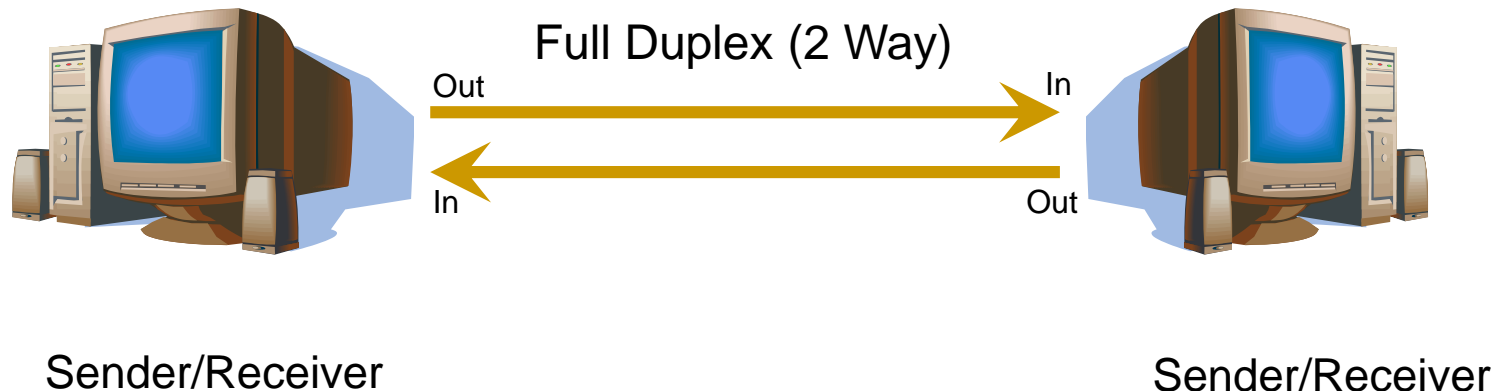


# Modes of Communications



## Full Duplex Communications (Bi-directional)

- Data can travel in both directions simultaneously.
- There is no need to switch between transmit and receive modes.
- Its like a two lane bridge or a two-lane highway.
- This type of communication requires only 3 wires, a ground, a transmit and a receive signal.
- Fire wire and [RS-232](#) use full duplex communications as well as simple digital serial comms.



# Universal Asynchronous Receiver/Transmitter(UART)



- Most **UARTS** are **full duplex**
  - They have separate pins and electronic hardware for the transmitter and receiver sections of the device that allows **serial output** and **serial input** to take place at the same time.
- Operation is based around **shift registers** and a **clock** signal.
- A UART clock (often referred to as the **baud rate generator**) determines shift register speed and hence **data rate** (bps)
- Most UARTs can also generate **parity** bits for transmission and perform parity checking on reception to provide simple error detection.

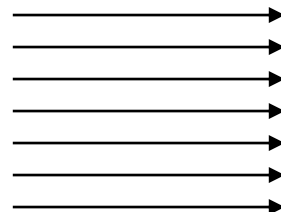
# UART - Transmitter



- Transmitter (Tx) - converts **parallel** data to **serial** data
  - Inserts **start** and **stop** bits used to indicate start and end of transmission.
  - Calculates and inserts **parity** bit if required into transmission.
  - Data rate is determined by the UART transmitter clock.

Status information back to computer  
(e.g. ready or error etc.)

Parallel data  
written by  
Computer

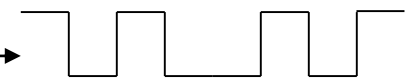


UART

Serial output

0-5v  
digital signal

0's and 1's



time

Transmitter Clock - from  
a baud rate generator  
circuit



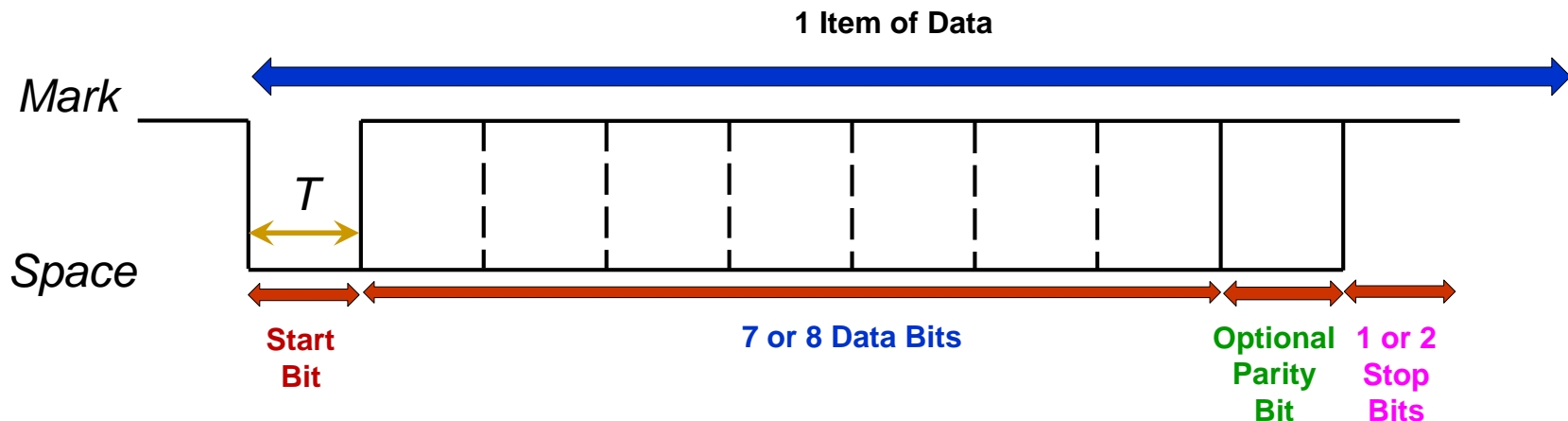
# RS232 Data Transmission and Framing



- When no data is being transmitted, the RS232 transmit signal remains at the IDLE (or Mark) level – Logical 1 (but actually -12v).
- Before a character can be transmitted it must first be 'framed'. The format of a typical character frame is as follows:-

- A Start bit:** logical 0 (also called a space +12v) indicates the start of a character.
- 7 or 8 Data bits**
- An Optional Odd/Even/No Parity bit** used for possible error checking.
- 1 or 2 Stop bits** at logical 1 (called a mark) that is used to indicate the end of the transmission of that item of data.

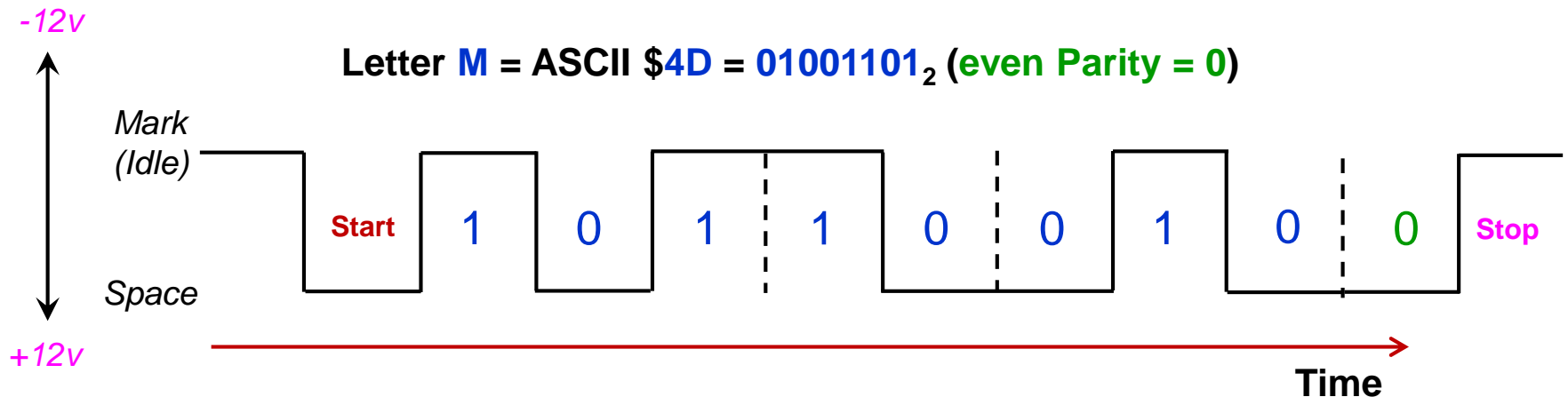
*(Two stop bits were used with old teletypes to give the mechanicals the opportunity to recover and prepare for the next characters. These days, 1 stop bit is common).*



# RS232 Data Transmission and Framing



- Given these combinations, there are **12 possible formats** for a character frame. Both transmitter and receiver have to agree the format.
- The illustration here shows how the letter 'M' would be framed and transmitted. Notice how the least significant bit is transmitted first, i.e. *little Endian*.



- As you can see, Asynchronous transmission is highly inefficient since with 8 data bits, a single start, stop and parity bit, we have only **73% data efficiency**

# Problem



## Exercise

- Draw the RS-232 bit pattern when transmitting the last three digits of your student number in ASCII.
- Assume the serial port is configured as 8-bits, 1 stop bit, even parity.
- How long does it takes to transmit these three digits at a clock speed to 9600 hz?

Character	ASCII
0	\$30
1	\$31
2	\$32
3	\$33
4	\$34
5	\$35
6	\$36
7	\$37
8	\$38
9	\$39

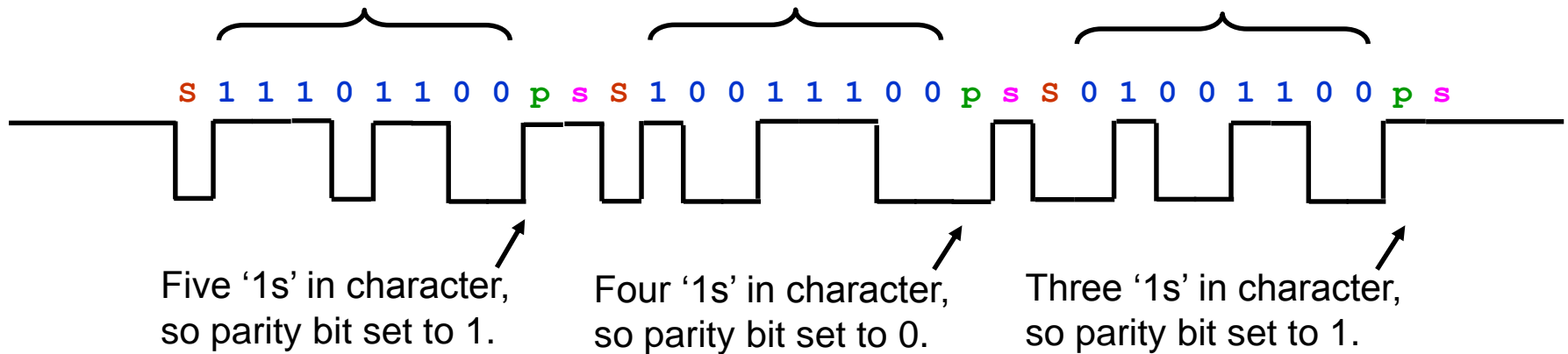
# Solution



'7' = 00110111

'9' = 00111001

'2' = 00110010



It takes  $3 \times 11 \times (1/9600) = 3.4375\text{ms}$  to transmit the three digits.

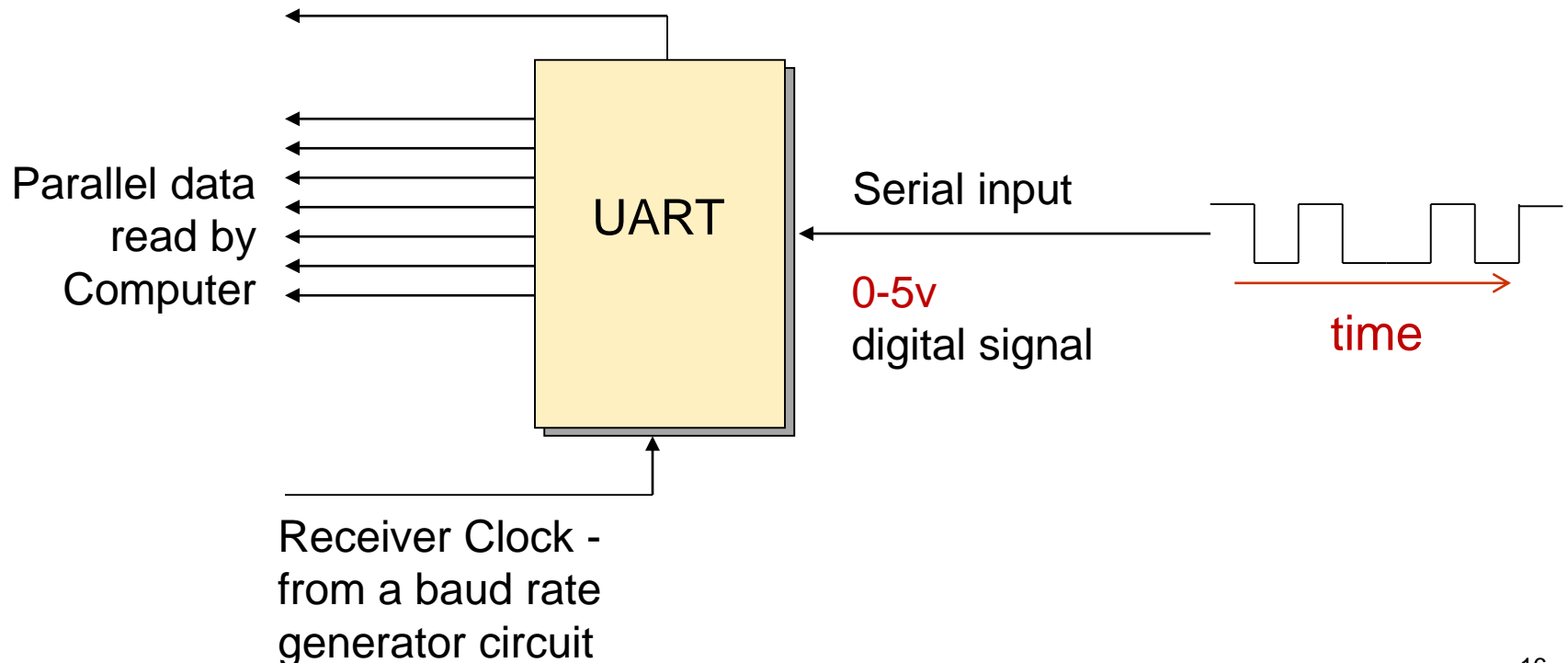
# UART - Receiver



## Receiver (Rx) Converts Serial Format Data back to Parallel

- Rx synchronises to transmitter using the **start** bit.
- Calculates **parity** and **checks** against **received parity bit**.

Status information back to computer  
(e.g. ready or error etc.)

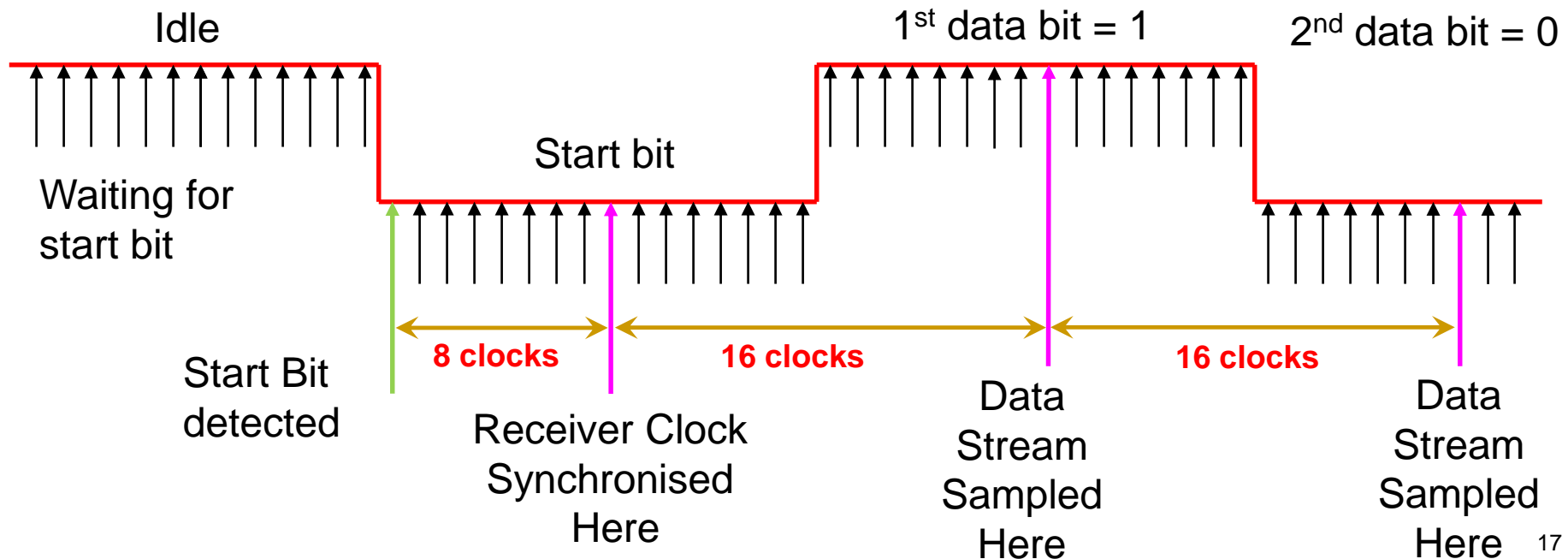




# Synchronising Receiver to Incoming Data



- How is data reception guaranteed, given that **RS-232** does **not** contain a clock that sender and receiver can synchronise too?
- **Receiver** samples the incoming data stream **16x's** faster than data (bit) rate.
- As soon as the **start bit is identified**, receiver **synchronises** it's shift register to the middle of 1<sup>st</sup> data bit time period, (i.e. **8 clock periods after detection of start bit.**) and thereafter samples each data bit every 16 clock periods.
- This means sender and receiver clocks can differ by as much as 2% (*easily achieved using crystal oscillators*) and still not drift to the point where the wrong data is sampled.

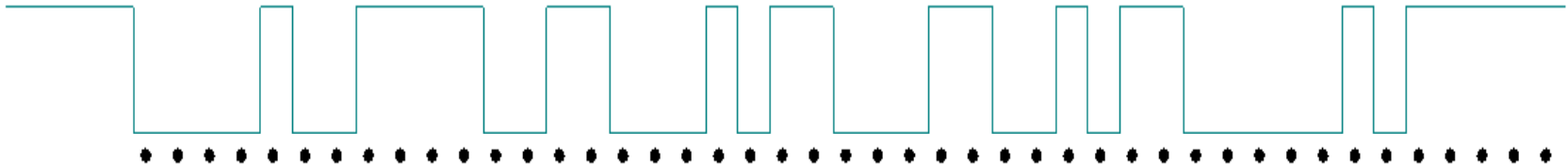




# Exercises

## Homework

- The bit stream below shows a RS-232 message transmitted with 7 data bits, odd parity, and 2 stop bits.
- Decipher the message assuming the characters transmitted are ASCII.
- **Remember** least significant bits transmitted first.

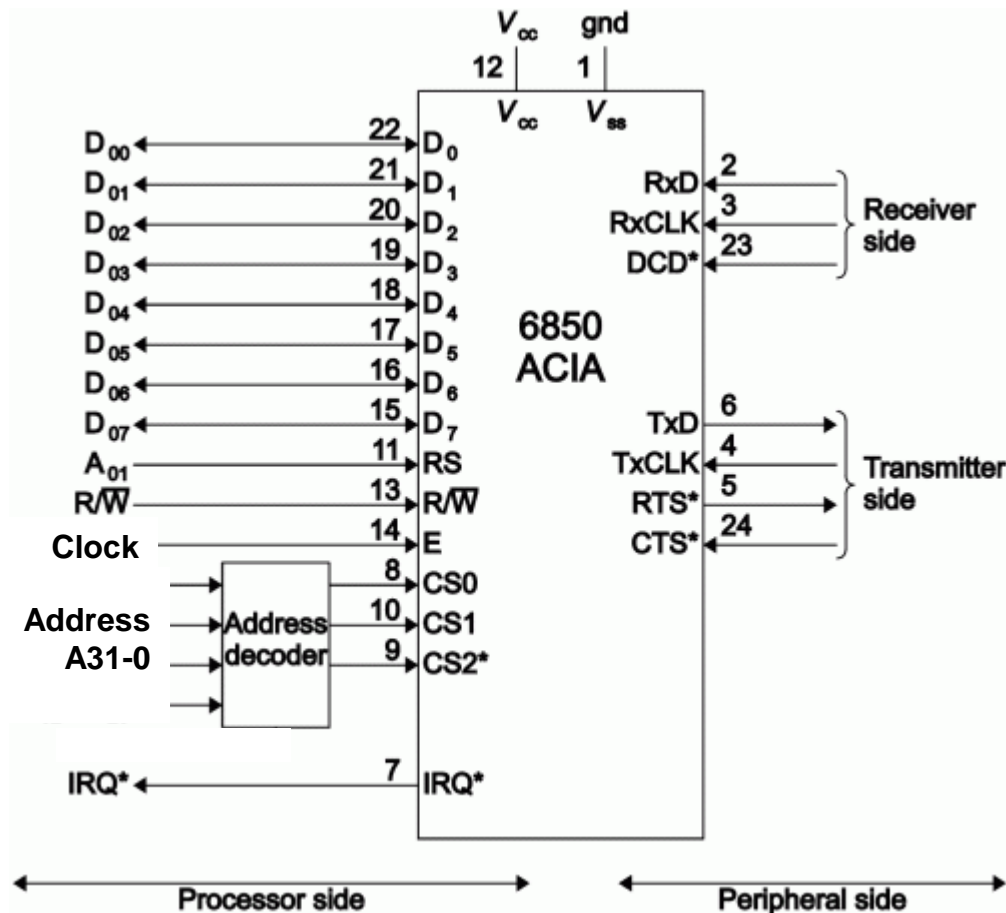


# The Motorola 6850 ACIA (UART)



## Introduction, History and External Interface

- The Motorola 6850 ACIA (a kind of UART) was one of the first single chip devices to handle the burden of asynchronous serial data.
- It has been rewritten in VHDL and is available on OpenCores.org.

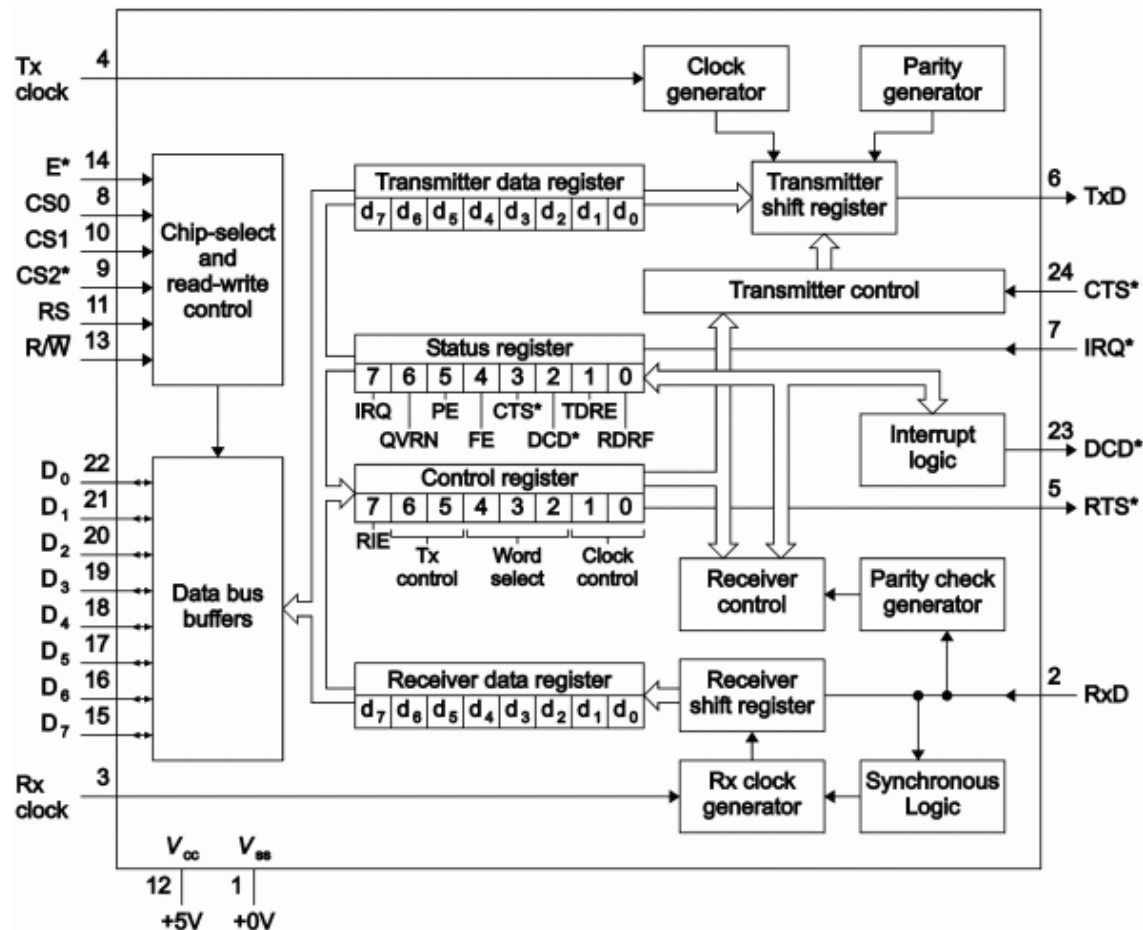


# The Motorola 6850 ACIA



## 6850 Internal Architecture - 4 byte wide Registers:

- **Control + Status** at address **hex 00400040**
- **Transmit + Receive Data** at address **hex 00400042**



# The Motorola 6850 ACIA



## The 6850 Control Register (write only)

- **Bits 0-1: Reset and Clock divisor**
  - Provide a software master reset for the device.
  - Clock division ratio (1, 16 or 64) for the transmitter and receiver.
- **Bits 2-4: Frame Format**
  - Data width (7 or 8 bits),
  - Parity (odd, even, none),
  - Number of stop bits (1 or 2).
- **Bits 5-6: Transmitter Control**
  - The RTS line used with a modem.
  - Whether the transmitter should generate an interrupt when the last character has been sent.
- **Bit 7: Receiver Control**
  - Determines whether the 6850 should generate an interrupt when a character is received.

BIT	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Function	Receiver interrupt enable	Transmitter control		Word select			Counter division	

CR1	CR0	DIVISION RATIO
0	0	1
0	1	16
1	0	64
1	1	Master reset

CR4	CR3	CR2	WORD SELECT			
			DATA WORD LENGTH	PARITY	STOP BITS	TOTAL BITS
0	0	0	7	Even	2	11
0	0	1	7	Odd	2	11
0	1	0	7	Even	1	10
0	1	1	7	Odd	1	10
1	0	0	8	None	2	11
1	0	1	8	None	1	10
1	1	0	8	Even	1	11
1	1	1	8	Odd	1	11

CR6	CR5	TRANSMITTER CONTROL	
		RTS*	TRANSMITTER INTERRUPT
0	0	Low (0)	Disabled
0	1	Low (0)	Enabled
1	0	High (1)	Disabled
1	1	Low (0)	Disabled and break

CR7	RECEIVER INTERRUPT ENABLE
0	Receiver may not interrupt
1	Receiver may interrupt

# The Motorola 6850 ACIA



## The 6850 Status Register (Read only)

- **Bits 0,1 (RDRF, TDRE)**
  - Bit 0 set to logic 1, if the receive data register is holding an unread character.
  - Bit 1 set to logic 1 when the transmit data register is empty.
  - Polling algorithms can check the status of these bits to determine if new characters have been received or previous ones transmitted.
  - Bit 0 is cleared when the receive data register is read from
  - Bit 1 is cleared when the transmit data register is written to.
- **Bit 2 (DCD)**
  - This bit reflects the status of the hardware modem control signal 'Data Carrier Detect' (DCD). It is used to inhibit reception of data unless an associated modem has detected a carrier signal.
- **Bit 3 (CTS)**
  - This bit reflects the status of the hardware modem control signal 'Clear to Send'. An active low signal on the 6850 CTS line indicates that some connected device is ready to receive more data from the 6850. This is a handshake signal in effect saying to the 6850 that it is clear to transmit more data.

BITS	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Function	IRQ	PE	OV RN	FE	CTS	DCD	TDRE	RDRF

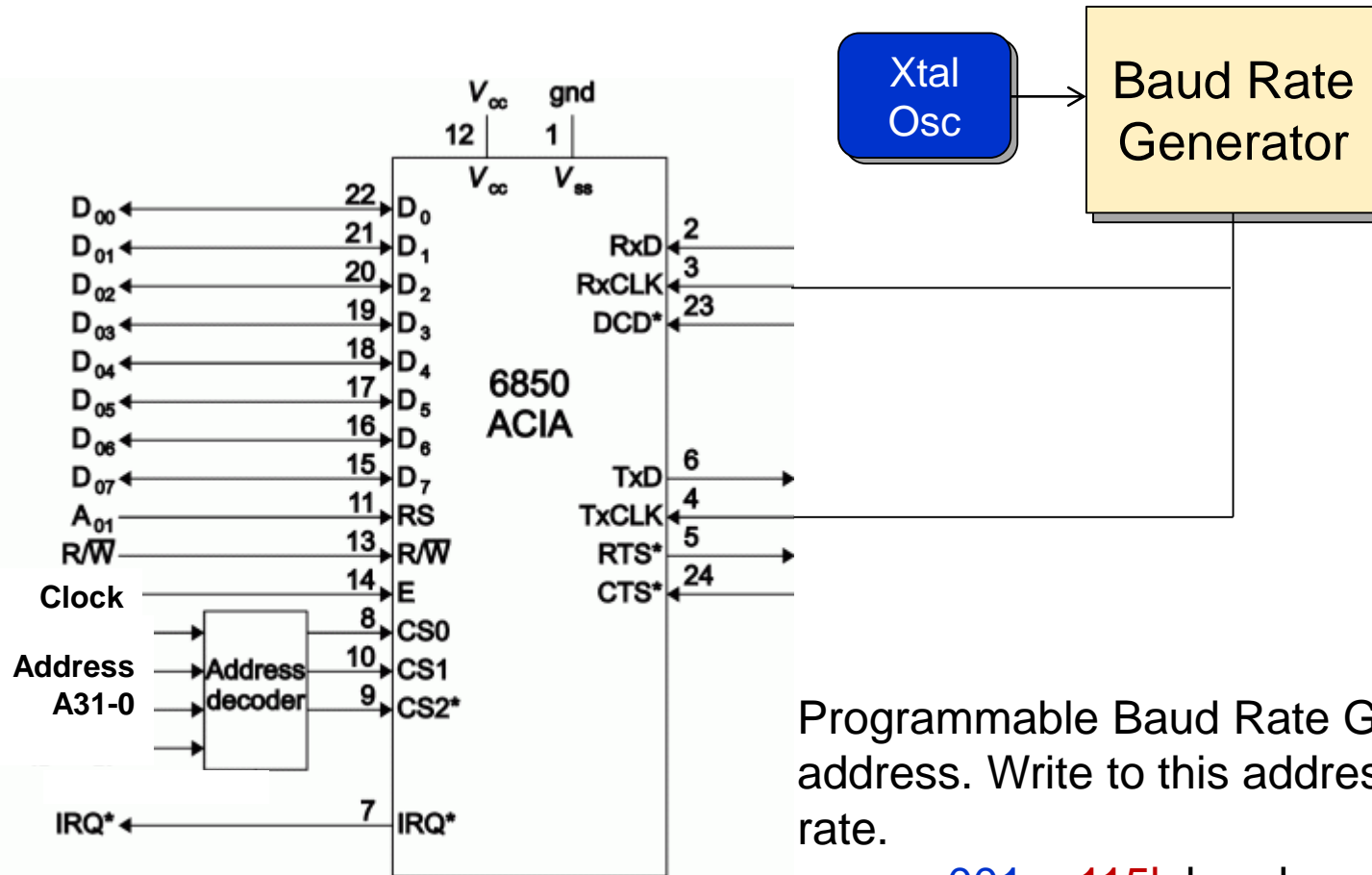
# The Motorola 6850 ACIA



- **Bit 4 (FE)**
  - This bit is set whenever the 6850 detects that a received character is incorrectly framed. Examples include reception of a character with the incorrect number of stop bits. This bit is set/cleared on a character by character basis.
- **Bit 5 (OVRN)**
  - This bit indicates that the 6850 has been overrun by received characters that were not read by the CPU. The implication being that data has been lost. This bit is set/cleared on a character by character basis.
- **Bit 6 (PE)**
  - This bit is set whenever the 6850 detects that a received character has Incorrect Parity. This bit is set/cleared on a character by character basis.
- **Bit 7 (IRQ)**
  - This bit is set whenever the 6850 wishes to interrupt the CPU. Typically this occurs when transmit buffer empty or receive buffer full interrupts have been enabled in the control register and is also set when Bit 2 (DCD) is set

BIT	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Function	IRQ	PE	OVRN	FE	CTS	DCD	TDRE	RDRF

# The Baud Rate Generator



Programmable Baud Rate Generator at address. Write to this address to set baud rate.

- 001 = 115k baud
- 010 = 57.6k baud
- 011 = 38.4k baud
- 100 = 19.2 baud
- all others = 9600 baud



# The Motorola 6850 ACIA



## 68000 Assembly Language Routines to Initialise 6850 (addresses are just examples)

```
ACIAControl    equ    $00400040
ACIAStatus     equ    $00400040
ACIATransmit   equ    $00400042
ACIAReceive    equ    $00400042
BaudRateGen     equ    $00400044
```

\*\*\*\*\*

- \* Initialise 6850 and baud rate generator for 115k Baud
- \* clock (16x), no RX/TX interrupts, RTS set low, 8 bit data, no parity, 1 stop bit
- \* All registers preserved

\*\*\*\*\*

```
Initialise    move.b    #%00000001, BaudRateGen    ; Program Baud rate for 115k
              move.b    #%00000011, ACIAControl    ; reset ACIA
              move.b    #%00010101, ACIAControl    ; x16 clock, 8 data, no parity, 1 stop, no interrupt
              rts
```

# The Motorola 6850 ACIA



## 68000 Assembly Language routines to Poll for character received

```
Control      equ      $00400040
Status       equ      $00400040
Transmit     equ      $00400042
Receive      equ      $00400042
DataRateGen  equ      $00400044
```

\*\*\*\*\*

\* Read status register until bit 0 = 1, then read receive data register and  
\* return character in d0.b. All other registers preserved

\*\*\*\*\*

```
Getchar      move.b    Status,d0          ; read status register into d0
              and.b     #%00000001,d0     ; mask off bits 7-1, leaving bit 0
              cmp.b     #%00000001,d0     ; is bit 0 = 1?
              bne       Getchar           ; if not keep polling
              move.b    Receive,d0        ; read received character into d0
              rts
```

# The Motorola 6850 ACIA



## 68000 Assembly Language routines to Poll status prior to Transmitting

Control	equ	\$00400040
Status	equ	\$00400040
Transmit	equ	\$00400042
Receive	equ	\$00400042
DataRateGen	equ	\$00400044

\*\*\*\*\*

\* Transmit Data

\* Character to transmit is in d0.b, all registers preserved

\*\*\*\*\*

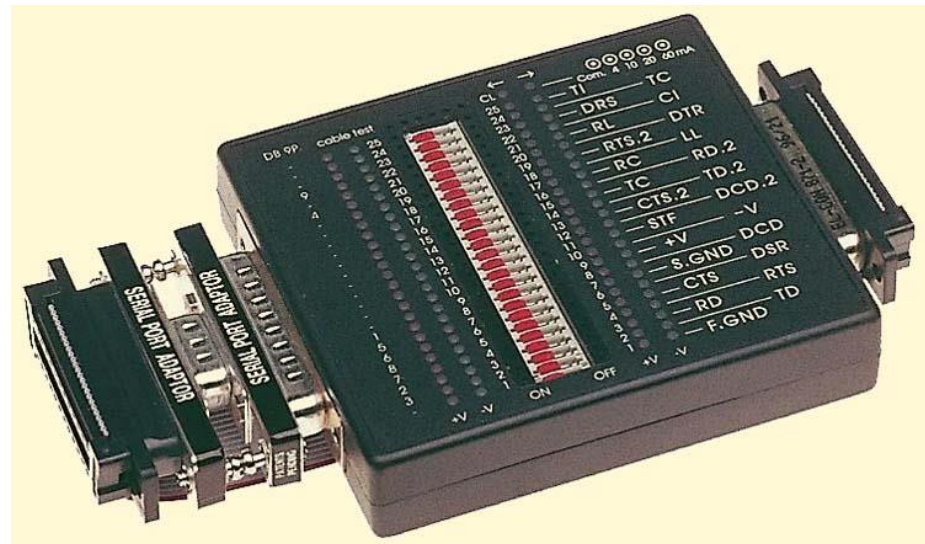
Putchar	move.b	d0,-(a7)	; push d0 into stack (save its value somewhere)
TxLoop	move.b	Status,d0	; read status register into d0
	and.b	#%00000010,d0	; mask off bits 7-2 and 0, leaving bit 1
	cmp.b	#%00000010,d0	; is bit 1 = 1?
	bne	TxLoop	; if not keep polling
	move.b	(a7)+,d0	; get character to transmit back off stack
	move.b	d0,Transmit	; write character to transmit register for transmission
	rts		



# The RS232 Standard

- The RS- 232 (Recommended Standard - 232) was initially published in the 60's.
- Currently the standard has been '[hacked about](#)' probably more than any other standard in existence for use [way beyond](#) its original applications, such as computer-to-computer or computer-to-printer applications.
- More time has been spent by Engineers attempting to make computer equipment communicate on RS-232 than just about anything else in existence !!!
- It has become so tricky connecting incompatible equipment together that '**breakout**' boxes are now sold so that technicians can play around with connecting one signal to another and observe the state of each signal on LEDs in order to establish communications (see illustration)

RS232 "Break-out"  
Box



# ***The RS232 Standard***

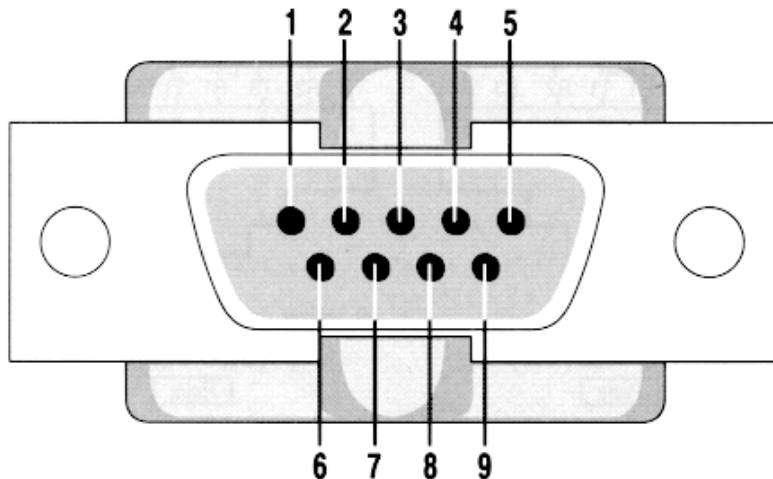


- **The RS232 Standard defines the following**
  - A Mechanical Specification
  - A Protocol to control the transmission of data using handshaking
  - An Electrical Specification

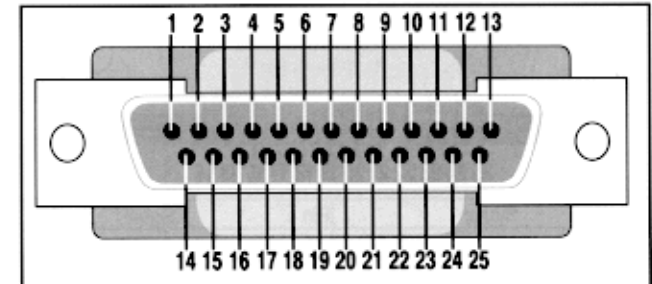


# The RS232 Mechanical Standard

- The standard specifies that equipment be connected via a **25 pin D-Type** connector (**DB-25**), although a **9 pin** connector (**DB-9**) is more often employed these days for its associated space and cost savings.
- The **Female** connector is generally (but not always) associated with **DCE** equipment (e.g. Modems, Printers, Terminals) while the **Male** connector is generally present on **DTE** equipment (computers etc.)



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		



Pin	Description	EIA CKT	From DCE	To DCE
1	Frame Ground	AA		
2	Transmitted Data	BA		D (Data)
3	Received Data	BB	D	
4	Request to Send	CA		C (Control)
5	Clear to Send	CB	C	
6	Data Set Ready	CC	C	
7	Signal Gnd/Common Return	AB		
8	Rcvd. Line Signal Detector	CF	C	
11	Undefined			
12	Secondary Rcvd. Line Sig. Detector	SCF	C	
13	Secondary Clear to Send	SCB	C	
14	Secondary Transmitted Data	SBA		D
15	Transmitter Sig. Element Timing	DB	T (Timing)	
16	Secondary Received Data	SBB	D	
17	Receiver Sig. Element Timing	DD	T	
18	Undefined			
19	Secondary Request to Send	SCA		C
20	Data Terminal Ready	CD		C
21	Sig. Quality Detector	CG		C
22	Ring Indicator	CE	C	
23	Data Sig. Rate Selector (DCE)	CI	C	
23	Data Sig. Rate Selector (DTE)	CH		C
24	Transmitter Sig. Element Timing	DA		T
25	Undefined			

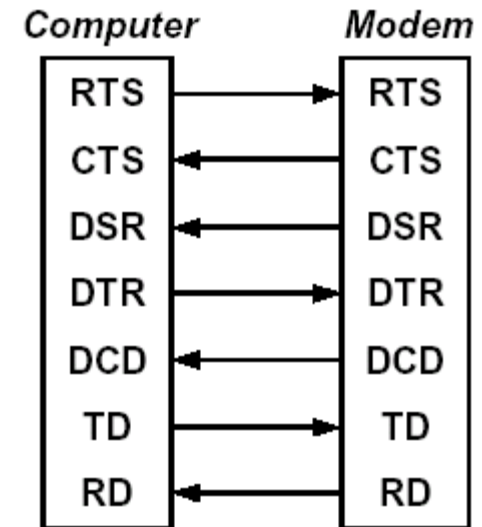
# The RS232 Protocol for Information Exchange



- **Request To Send (RTS), Pin 4**
  - Asserted by the computer to inform the modem that it wants to transmit data.
- **Clear To Send (CTS), Pin 5**
  - Asserted by the modem after receiving a RTS signal, indicating that the computer can now transmit.

Control Flow  
Signals  
(Handshaking)

- **Data Terminal Ready (DTR), Pin 20 (DB25) or Pin 4 (DB9)**
  - This signal is asserted by the computer, to inform the modem that the computer is ready to communicate
- **Data Set Ready (DSR), Pin 6**
  - This signal is asserted by the modem in response to a DTR signal from the computer and indicates the modem is ready to communicate.
- **Data Carrier Detect (DCD), Pin 8**
  - This signal is asserted by the modem, informing the computer that it has established a physical connection with the telephone network.



Status Signals

- **Transmit Data (TD), Pin 2**
  - This signal carries the transmitted data from the computer to the modem.
- **Receive Data (RD), Pin 3**
  - The signal carries the received data from the modem to the computer.

Data Signals

# RS-232 Handshaking



## Handshaking

- Handshaking is a protocol whereby if the receiver cannot keep up with the data rate of the sender, the receiver can slow down the sender.
- There are two methods of handshaking. Most equipment supports both and is purely down to configuration or choice as to which to support

- Hardware: Using wires (RTS/CTS).
- Software: XON, XOFF



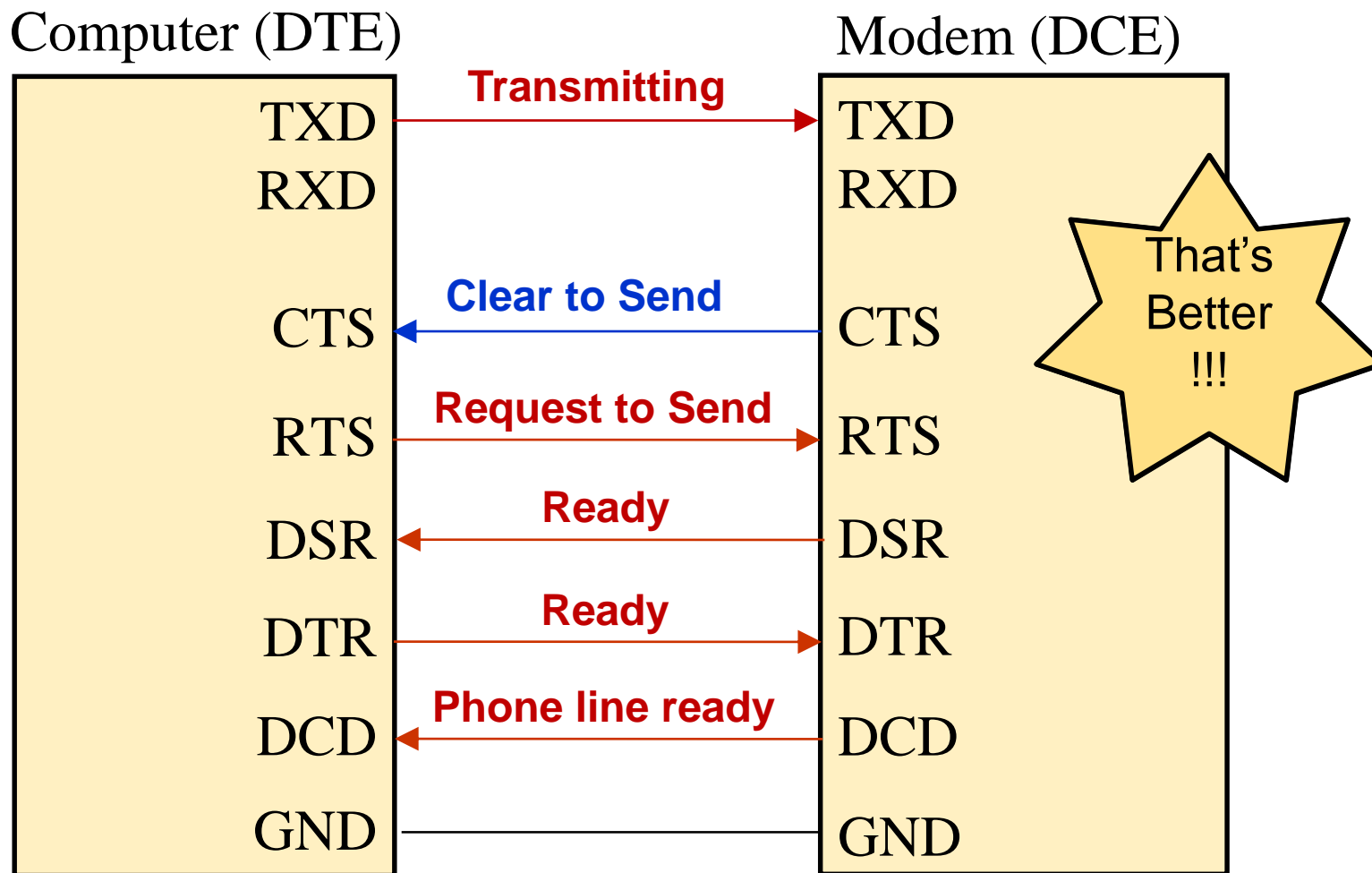
# RS-232 Handshaking



## RTS/CTS Handshaking

- If the sender is *transmitting data too fast* for the receiver to keep up, *i.e. the receivers data buffers are getting full and it cannot process that amount of data so quickly*, it will remove the **CTS** (Clear-to-send) signal, this will **disable** the sender from transmitting.
- The **6850s** transmitter section is hard wired to this signal so it will not transmit unless **CTS** (Clear-to-send) signal is active.
- Note handshaking works *only* in *one direction*, that is, that DCE equipment (e.g. modem, printer, terminal etc.) is slow and thus needs to can control the transmission speed of DTE equipment, e.g. computer.
- It is assumed that a computer (DTE) is fast enough to keep up with any minimal data back from say a printer, terminal or Modem (DCE). Two computers should be fast enough to keep up with each (given that physical communications over the RS232 is quite slow). However, Xon-Xoff can be used if they need to control each other

# RS-232 Handshaking



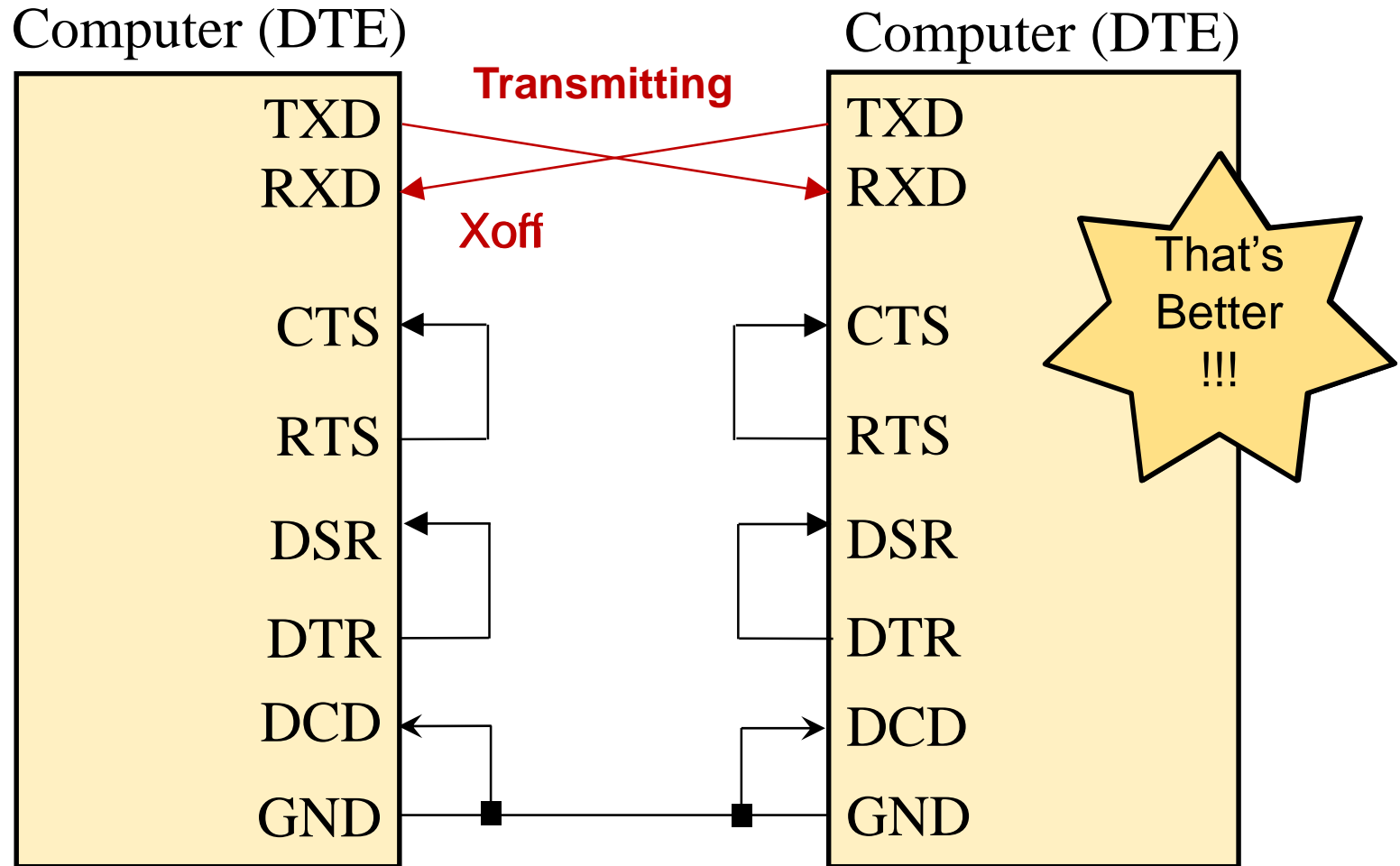
Configuration for Hardware Handshaking, e.g.  
Computer to Modem

# RS-232 Handshaking



- **Soft Handshaking using XON-XOFF**
  - In systems requiring the minimum of cabling complexity and cost, computers and peripherals can be connected without physical/electrical handshaking, that is CTS/RTS handshaking is disabled (or bypassed).
  - Instead, a software protocol (as opposed to an electrical/signalling protocol) is employed whereby two way handshaking is implemented using the transmission of characters XON (free to transmit) and XOFF (stop transmitting).
  - These characters correspond to hex 11 (Ctrl-Q) and hex 13 (Ctrl-S) from the ASCII chart.
  - This protocol is not defined by the RS-232 standard but is more commonly used than CTS/RTS signalling, particularly when implementing Computer-to-Computer communications.
  - In such situations, the CTS/RTS and DTR/DSR pins have to be wired together to permanently enable the communication devices.
  - The illustration here demonstrates the idea when using a 3 wire cable to connect DTE/DTE equipment.

# RS-232 Software Handshaking



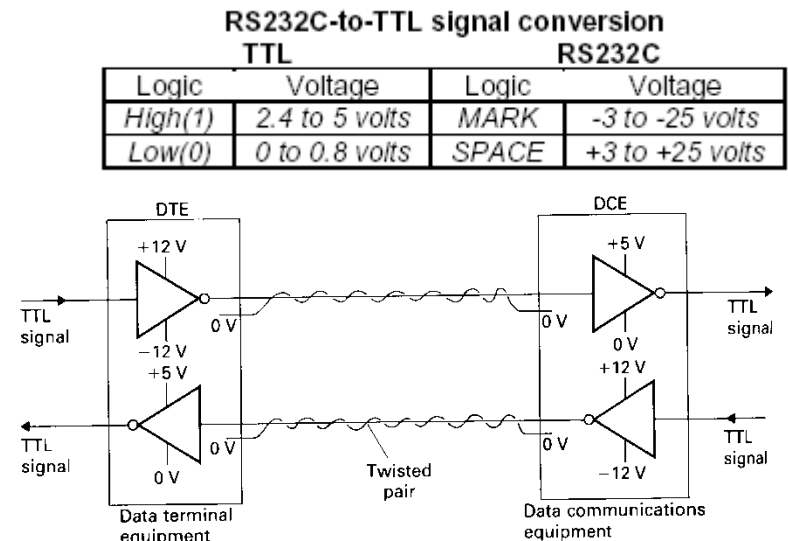
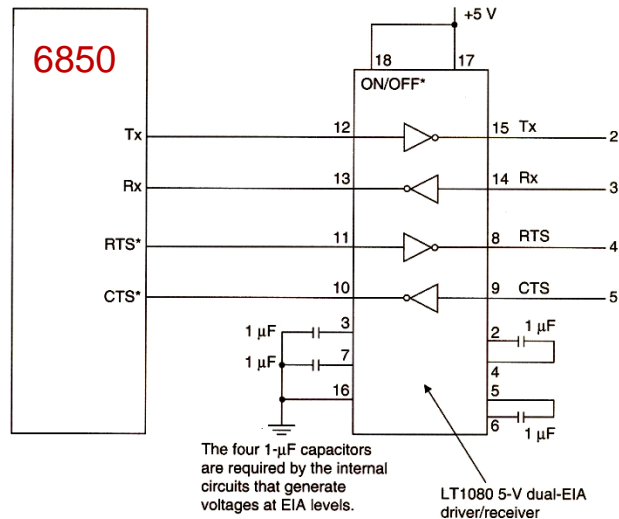
Configuration for Software Handshaking using minimal 3 wire connection **Tx**, **Rx** and **ground**, e.g. Computer to Computer (**Note** crossed over transmit and receive and CTS connected to RTS etc.)



# The RS232 Electrical Standard

## RS232 Electrical Interface

- All Signals are valid in the range  $\pm 3$  volts to  $\pm 25$  volts although  $\pm 12$ v is more common. Signals within the range  $-3$  volts to  $+3$  volts are considered **invalid**.
- **MAX** do a full range of RS-232 driver chips in various configurations. As do Linear Technologies, these convert digital (TTL voltage levels in the range 0-5 into RS-232 levels in the range  $\pm 12$ v.
- Both sets of chips use on-chip circuitry to generate RS-232 voltage levels from a single 5v supply.
- The example below uses Linear Technologies LT1080 chip.





## ***DTE to DTE connections and Null Modems***

- RS-232C interface designed originally to connect computers to modems
- However, many of today's RS-232C applications are for connecting a computer (**DTE**) to either another computer or to a peripheral such as a printer (also considered a **DTE**), rather than to a modem (**DCE**)
- In order to connect a computer to a computer (i.e. **DTE** to a **DTE**) we must **cross** over some of the control signals. This is where a break-out box can be useful !!
- This **DTE-DTE** interface is accomplished with a special connector cable called a *null modem*
  - A NULL modem simulates a **DTE-DCE/DCE-DTE** circuit
  - A NULL modem has a female connector on both ends.
  - The signals are wired as indicated in the figure overleaf.



# ***DTE to DTE connections and Null Modems***



- NULL Modem Connections for Computer to Computer Communication
- Note how everything is crossed over
- Handshaking Can be implemented using [RTS/CTS](#) or [Xon-Xoff](#).

