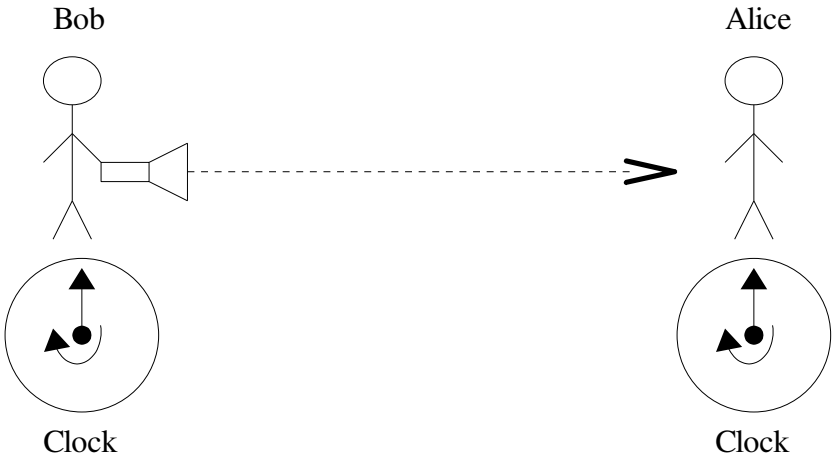
Serial communications.
Serial communication involves the sending of data between transmitters and receivers one bit at a time. This might seem a very slow way to send data however it is often cheaper and simpler to run 1 data wire instead of say 16. In other cases, it is easier to send data quickly one bit a time a time instead of 8 or more bits together. As to speed, we are now witnessing the use of serial ATA hard disks which are achieving a higher interface data rate than their older parallel ATA counterparts. Sending data in parallel may seem faster on the face of it however, at high speed, it is not likely that all of the bits will travel at the same rate along the data cable (or PCB tracks). For this reason, data rates must be limited so that the mismatch in arrival times of the bits is not significant compared to the overall read/write cycle time. Serial transmission does not suffer from this limitation.

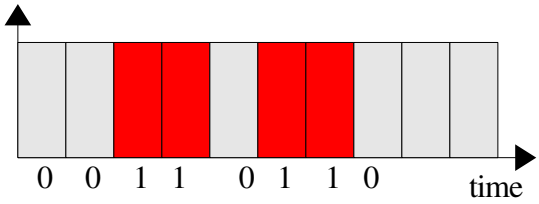
For the purposes of this document we will treat serial communications as if it falls into two main categories: Synchronous and Asynchronous. We will begin by looking at Asynchronous serial data transfer.

Asynchronous serial communications.

Bob has a flashlight and wants to send the 8 bit number 0011 0110 to Alice. They have previously arranged that data will be sent at a rate of 1 bit per second starting every 8 seconds. Bob therefore waits for the start of a 8 second "window" and begins sending data. Alice, watching the clock, notes the start of a 8 second window and starts looking out for flashes.



Bob sends

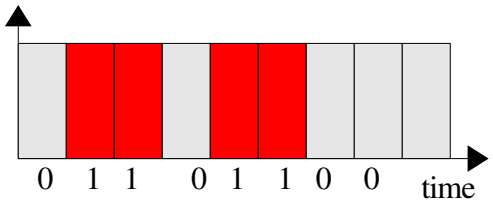


Alice watches the clock and the flash-light and concludes that Bob sent 00110110 (The extra two 0's at

the end are simply there because they are the first two bits of the next byte to be sent).

If this is to work properly three things are required.
Both clocks must be synchronised to the same 8 second window.
Both clocks must run at the same rate.
There should not be too much interference or distortion of the light signal.

Suppose Alice's clock is 1 second slow. Instead of seeing what Bob intended, she will see this:



So she will think that Bob has sent 0110 1100. A similar error arises if the clocks run at different rates.

In practice, asynchronous serial communications devices usually send a synchronising signal to synchronise the clocks. In the case of RS232/422/485 – the most common examples of asynchronous serial communications, each byte is preceded by a "Start-bit", the purpose of which is to resynchronise the clocks at either end. A "Stop-bit" is also sent to confirm the end of transmission of the byte. The clocks used at either end are also usually derived from crystal controlled oscillators. This makes them very accurate and so they can usually be relied upon to stay synchronised between start bits.

The transmission system described uses 2 bits (at least) as an overhead to send 8 bits of data which represents a fairly large management overhead. The reason this overhead is present is to correct for drift between the sender's and receiver's clocks. This drift can be eliminated if both sender and receiver share the same serial clock.

Synchronous serial communications.
Synchronous serial communications occurs when the sender and receiver are synchronised to the same clock source. The easiest way to achieve this is to simply include the clock signal in the transmissions. This can be done using a separate wire or by embedding the clock signal in the data stream itself. High speed data communications such as Ethernet and Serial ATA both use synchronous serial communications.