Serial Port Communication

Babak Kia
Adjunct Professor
Boston University
College of Engineering
Email: bkia -at- bu.edu

Why Serial Port?

- **Serial ports are one of the simplest interfaces to get working on a microprocessor**
- **As a result, most development boards come equipped with one, and most developers start by getting the serial port to work first**
- **Most host PCs are also equipped with at least one serial port, and this makes interfacing to the microprocessor a simple task**

Definitions

- ***Serial Communication* occurs when data is transmitted bitwise and in a sequential manner over a single wire.**
- ***RS232* is a physical interface standard for the interconnection of devices, carrying signals between ±5v and ±12v, as defined by the Electronics Industry Association (EIA). Also referred to as EIA RS-232C**
- **A *UART*, or a Universal Asynchronous Receiver/Transmitter is the IC component which manages the transmission and reception of serial data. It converts the serial data to parallel data which can be used by the CPU.**

Definitions

- A ***half-duplex*** **connection is a communications channel where both transmission and reception are possible, but only in one direction at a time**
- **A *full-duplex* connection permits the sending and receiving of data at the same time**
- ***Baud*** **is the transmission rate at which data communications occurs. It is roughly equal to the number of bits per second. Named for JME Baudot, the inventor of the Baudot telegraph code.**
- ***MARK*** **is the idle state of the signal, and negative with respect to the common**
- ***SPACE*** **is the active state of the signal, and positive with respect to the common**

Serial Communication Interface

- **The Serial Communication Interface (SCI) term was coined by Motorola in the 1970s to identify a UART.**
- **The SCI contains both parallel-to-serial and serial-to-parallel converters, alongside interrupt capabilities and multiple transmit/receive modes**
- **Other features include parity checking, error detection, programmable character lengths, and a varying number of stop bits**

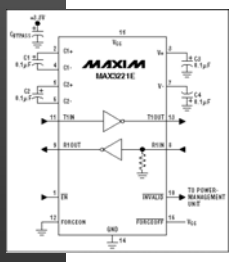
Signal Names

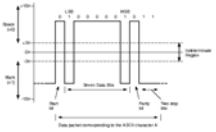
- **CTS – Clear To Send**
  - **Asserts (logic '0', positive voltage) when the device is ready to receive a transmission.**
- **RTS – Request To Send**
  - **Asserts (logic '0', positive voltage) when the device is ready to send a message. This can be used to enable a receiver, or be used in half-duplex mode**
- **RxD – Receive Data**
  - **Active when the device receives a signal, the idle state defaults to a Mark state (logic '1', or a negative voltage)**
- **TxD – Transmit Data**
  - **Active when a signal is being transmitted, the idle state defaults to a Mark state (logic '1', or a negative voltage)**

RS232

- **RS232 uses**
  - **+3 to +12 volts to indicate an ON (SPACE) state**
  - **-3 to -12 volts to indicate an OFF (MARK) state**
- **The "dead area" between -3v and +3v is to prevent noise from affecting communications**
- **Microprocessor systems which operate on 3 or 5 volts don't need to provide an extra power source to power the -12 and +12v RS232 circuitry. RS232 chips generate these voltage using a single 3 or 5 volt power source**

Transmitting a byte





Timing

- **There are two sets of timings to keep in mind**
  - **Baud rate, which sets the communications speed**
  - **Inter-byte delay. Some devices, even though they are capable of operating on high baud rates may not be able to accept too many characters back-to-back. This is due to either a small queue size or being busy with other interrupts.**
  - **In order to avoid an overflow error, either use hardware handshaking, or introduce an inter-byte delay**