Data Acquisition & Control: Serial Communication

Janusz Zalewski
Florida Gulf Coast University
Outline

- Serial Communication in General
- RS232C
- Java Libraries
Overview

- Serial communication means transmitting data **one bit** at a time versus **one byte** at a time in parallel comm.
- In principle one only needs one signal line (wire) and one return wire, which should be inexpensive.
- However, serial comm tends to be more complicated & expensive due to the lack of strictly defined standards.
- One well known standard exists, RS232C, which is the basis for a Serial Port available in most computers.
Principal Transmission Options

- **Simplex**: One wire transmits, one wire receives.
- **Half Duplex**: Either unit can transmit while the other receives but both cannot transmit simultaneously.
- **Full Duplex**: Both can transmit simultaneously.
Transmission Techniques

- In asynchronous transmission, information is sent at random intervals. The device sends a start and stop bit associated with each transmission, so the synchronization could be achieved.
- In synchronous transmission, timing information continuously sent across the wires connecting the two devices. So no start and stop bit is required. This method is more efficient but is mainly used in mainframe computer communications.
**RS232C Connector Types: DB9 and DB25**

### Table: RS232C Connector Types

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Name</th>
<th>Dir</th>
<th>Notes/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DCD</td>
<td>IN</td>
<td>Data Carrier Detect. Raised by DCE when modem synchronized.</td>
</tr>
<tr>
<td>2</td>
<td>RD</td>
<td>IN</td>
<td>Receive Data (a.k.a RxD, Rx). Arriving data from DCE.</td>
</tr>
<tr>
<td>3</td>
<td>TD</td>
<td>OUT</td>
<td>Transmit Data (a.k.a TxD, Tx). Sending data from DTE.</td>
</tr>
<tr>
<td>4</td>
<td>DTR</td>
<td>OUT</td>
<td>Data Terminal Ready. Raised by DTE when powered on. In auto-answer mode raised only when RI arrives from DCE.</td>
</tr>
<tr>
<td>5</td>
<td>SGND</td>
<td>-</td>
<td>Ground</td>
</tr>
<tr>
<td>6</td>
<td>DSR</td>
<td>IN</td>
<td>Data Set Ready. Raised by DCE to indicate ready.</td>
</tr>
<tr>
<td>7</td>
<td>RTS</td>
<td>OUT</td>
<td>Request To Send. Raised by DTE when it wishes to send. Expects CTS from DCE.</td>
</tr>
<tr>
<td>8</td>
<td>CTS</td>
<td>IN</td>
<td>Clear To Send. Raised by DCE in response to RTS from DTE.</td>
</tr>
<tr>
<td>9</td>
<td>RI</td>
<td>IN</td>
<td>Ring Indicator. Set when incoming ring detected - used for auto-answer application. DTE raised DTR to answer.</td>
</tr>
</tbody>
</table>

### Diagrams:

- [DB9 Connector Diagram](image1)
- [DB25 Connector Diagram](image2)
Signal Levels & Modulation

- TTL signal levels are generally too low voltage and too sensitive to noise to be used over longer distances, so serial communication uses higher voltage differential +12V and -12V.
- Many modulation schemes exist:
  - Voltage level to determine logical 1 or 0
  - Negative transition between levels to be 0
  - Positive transition between levels to be 1
Parity bit sent in serial communication

- Sent primarily when ACSII data is transmitted. Permits limited error checking.
- A single bit is added to the transmitted character so that the sum of the bits transmitted is even or odd parity.
- The receiving unit can determine if the data have the desired parity and determine if an error has occurred during the transmission process.
Serial Communication Configuration

- **Baud Rate:** typically 9600, normally ranges from 300Hz to 112,000Hz
- **Parity:** (Y/N)
- **Bit Length:** number of data bits (7/8)
- **Stop Bits:** number of stop bits (1/2)
- **Common configuration string:** 9600,N,8,1
Java Serial Libraries

- RXTX: Open source java serial port library:
  

- SerialPort: Commercial java serial port library:
  
Programming Example

- Get local serial port names
  - names = SerialPortLocal.getPortList();
- Assign Port Object to first system serial port
  - devName = args[0];
    SerialPortLocal.addPortName(devName);
- Create serial port object.
  - SerialConfig serCfg = new SerialConfig(devName);
    serCfg.setBitRate(SerialConfig.BR_9600);
    sp = new SerialPortLocal(serCfg);
    sp.setDTR(true); //some modems will not respond if DTR is low
Programming Example

- Create and start threads for sending and receiving data
  - TermRcvTask rcv = new TermRcvTask(sp);
  - TermSndTask snd = new TermSndTask(sp);
  - rcv.start();
  - snd.start();
Receiving Thread

```java
class TermRcvTask extends Thread {
    TermRcvTask(SerialPortLocal sp) throws IOException {
        if (sp.getPortNum() == -1)
            throw new IOException("TermSndTask: serial port not initialized");
        p = sp;
    }
    public void run() {
        int b;
        // System.out.println("TermRcvTask running...");
        for (;;) {
            try {
                System.out.println("p.getByte()...");
                while((b = p.getByte()) != -1) {
                    if (b < 32 || b > 126)
                        //System.out.print("<"+b+">");
                        System.out.print("");
                    else
                        System.out.print((char)b);
                }
                try {Thread.sleep(100);} catch (InterruptedException e){}
            } catch (Exception e) {
                System.out.println("Error in TermRcvTask "+e);
            }
        }
    }
    SerialPortLocal p;
}
```
Sending Thread

class TermSndTask extends Thread {
    TermSndTask(SerialPortLocal sp) throws IOException {
        if (sp.getPortNum() == -1)
            throw new IOException("TermRcvTask: serial port not initialized");
        p = sp;
    }
    public void run() {
        int b;
        for (;;) {
            try {
                System.out.println("Input...");
                b = System.in.read(); // blocks if input not available
            System.out.println("b="+(b+" ");
                if (b == 10) p.putByte((byte)13);
                else p.putByte((byte)b);
            } catch (Exception e) { System.out.println("Error in TermSndTask "+e); }
        }
    }
    SerialPortLocal p;
}