Serial Communications

A brief history and it's use in the side4linux Project.

Table of Contents

Serial communications is the sending of binary data such as letters, numbers and punctuation symbols as a series of logic true and logic false bits (one bit at a time) corresponding to some code that represents these symbols. We will briefly consider the history behind serial transmissions so as to understand some of the jargon describing the RS232C serial interface. The RS232C serial interface is used to enable communication between the controlling Linux computer and our Machine Controller.

Telegraph System

The first successful long distance serial communication system was the overland telegraph invented by Samuel F.B. Morse who devised his own Morse Code to be transmitted using a switch to switch electrical current on and off at the transmitting end of a wire circuit terminated by an electromagnetic sounder. The sounder created a noise at the receiving end that the receiving operator could then hear and decode back into the text message that was originally sent. Unfortunately it was left to the receiving operator to decode the Morse code back into a message so if the receiving operator decoded the message incorrectly then the original message could easily have been misunderstood. Morse Code is a series of triplets (three bits per character) which corresponded to the upper case alphabetic characters, numbers 0->9 and some basic punctuation. The logic state for each bit of a triplet code was determined by the length. Long was called a 'dash' and short was called a 'dot'. The telegraph system worked very well but had some shortcomings most notably the need for a trained operator at both ends, the very slow speed of transmission and because the wiring was direct, both the originator and recipient of the message had to either attend the Telegraph Office or send messengers.

Telegraph Sending:

`originator -> operator -> switch -> telegraph wires -> sounder -> operator -> recipient.`

Telegraph Receiving:

`recipient <- operator <- sounder <- telegraph wires <- switch <- operator <- originator.`

Telephone System

After the Telegraph came the Telephone System invented by Alexander Graham Bell which converted human voice directly into analogue[A] electrical signals which were transmitted through the wires into a Telephone Exchange and then on to the recipient where the electric current was converted back into voice. The use of this system became much more widespread than the telegraph system it had begun to replace due to it's immediacy and the convenience of having the transmitter and receiver in the home or office. The Telephone system is not useful as a serial data transmission system for equipment as it was designed only to allow human beings to talk directly to each other. As use of the Telegraph system declined there was still a need to transmit serial data over long distances by machine so the 'teletypewriter' came into use. This system was basically an electro-mechanical typewriter that generated five bits per character 'Baudot' code to match the key being pressed. The code was punched directly onto paper tape by the 'typewriter'. A Telephone connection would be made with the recipient and the originator would start up the punched tape reader which would then 'read' the taped message at high speed converting it into analog tones through a 'modulator'. The 'modulator' would send the message down the telephone wire to a 'de-modulator' at the recipient's end. The 'de-modulator' would read incoming tones and convert them back into digital electrical signals which could then be punched onto tape as 'Baudot' code by a hole punching machine. After disconnecting the Telephone the recipient would then feed the punched tape through another paper tape reader where the holes in the tape would be read and decoded into electrical signals to drive another electro-mechanical typewriter. By this means a text message could be created and sent vast distances over the telephone network, at high speed, in real time, and printed at the other end at leisure without the need of trained operators. At this point it should be understood that the typewriting and tape equipment became known as Data Terminal Equipment and the Telephone line driving and receiving modulators/de-modulators became known as Data Communication Equipment.

Teletype Sending:

```
digital -> DTE -> DCE -> Telephone Line ->

-> Telephone Exchange ->

Telephone Line -> DCE  -> DTE -> digital.
```

Teletype Receiving:

```
digital <- DTE <- DCE <- Telephone Line <-

<- Telephone Exchange <-

<- Telephone Line <- DCE  <- DTE <- digital.
```

A: Analogue is defined here as a signal varying in voltage level and frequency instead of an on and off digital signal.

Teletype System

Now the Teletype was and still is a tradename of the Teletype Corporation but all machines with this 'teletypewriter' function ended up being called 'teletypes'. The Teletype combines the function of reading and punching the paper tape with the electro-mechanical typewriter. The modulator/de-modulator also became one item that was called a 'Modem' (shorthand for MOdulator/DEModulator). The type of equipment was divided up into Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). The Teletype is DTE because it is the terminating device at both ends of the link and the Modem is the communication equipment (DCE) that connects the Teletype to the telephone line at both ends.

So to summarise the DTE sends a serial digital message to a Modem (DCE) and out onto the Telephone System as a series of analogue electrical signals representing sound tones. At the other end of the telephone connection a Modem (DCE) converts the analogue electrical signals representing sound tones back into digital which is then converted back into printed character data by another DTE.

Over time the speed at which the Teletype transmission was sent became known as the 'bit rate' measured in bits per second. Because of the use of the 'Baudot' coding system another measurement of the transmission rate became to be known as the 'baud rate' (a 'baud rate' of one is about one character per second). To reduce errors a system of Parity was introduced which basically means that if you are using Odd Parity another bit is added to the end of the character as a Mark to make the sum of bits of the character an odd number (Even Parity is also used). You could also declare that there was No Parity in which case the parity test would not be done.

After some time it became clear that the five bit per character Baudot code carried insufficient information for normal use (5 to the power of 2 equals only 32 characters) so the seven bit (128 characters) American Standard Code for Information Interchange (ASCII) came into widespread use. ASCII code provides upper and lower case characters, the numeral set of 0->9, punctuation characters, and a greater range of control codes useful in the serial transmission process.

There was still one problem to overcome with the Teletype system and that was standardising the signals which are used by the DTE and DCE to communicate their control states to each other. This process of communicating control state is often referred to as 'handshaking' and is done either with separate dedicated wiring or by passing software control codes back and forth (e.g. Xon/Xoff Protocol). In the beginning manufacturers provided the Teletype and the Modem and there was little in the way of standardisation which meant that anything could be expected in the area of handshaking between the two. Careful consideration was needed about which Modem to connect to which Teletype (some models from the same manufacturer even had problems!). At this time a new Standard came into being and is called the 'RS232C Standard'. This Standard defined the voltage levels for on and off (also called Mark and Space), the electrical impedance of the inputs and outputs and it defined the Handshaking lines and Software Control Protocols. One important thing to be considered now in this discussion is the definition of the electrical transmission signals. This means that using the RS232C Standard adds another item into the communication channel called the 'level shifter' or 'interface'.

The link to transmit a serial signal using the RS232C Standard is now as follows,

RS232C Sending:

```
digital -> DTE -> Level shift to RS232C -> Serial Wiring -> Level shift back to digital -> DCE ->
Telephone Line -> Telephone Exchange -> Telephone Line ->
-> DCE -> Level shift to RS232C -> Serial Wiring -> Level shift back to digital -> DTE ->  digital.
```

RS232C Receiving:

```
digital <- DTE <- Level shift back to digital <- Serial Wiring <- Level shift to RS232C <- DCE <-
Telephone Line <- Telephone Exchange <- Telephone Line <-
<- DCE <- Level shift back to digital  <- Serial Wiring <- Level shift to RS232C <- DTE <-  digital.
```

The need for the level shifters is that the RS232C standard sets the voltage for a Mark (logic true) as between +3 and +15 volts and the level for a Space (logic false) as between -3 and -15 volts. Between -3 and +3 volts is no man's land where the logic level does not change (this is called hysteresis) to prevent noise and short circuits creating false signals. As you can see there is no match to current digital voltages (e.g. True = +5V, False = 0V). This process is called signal level translation.

Handshaking

To assist RS232C communication between DTE and DCE a system of handshaking is employed. This interface consists of additional wires that convey the state condition of the two units to each other and/or the use of software handshaking protocols such as 'Xon/Xoff'. In this discussion we will only consider the hard wired handshaking. State information could be 'ready to receive data' or 'ready to send data' or a number of other conditions and are summarised below along with their electrical pin outs as related to modern PC and Modems equipped with DB9 nine pin sockets.

Remembering that the PC is DTE and the Modem is DCE,

DTE – Male DB9 Socket

Pin Number	I/O	Name	Abbreviation	Active
1	INPUT	Carrier Detect	CD	LOW
2	INPUT	Received Data In	RDI	
3	OUTPUT	Transmitted Data Out	TDO	
4	OUTPUT	Data Terminal Ready	DTR	LOW
5	GROUND	Signal Ground	GND	n/a
6	INPUT	Data Set Ready Input	DSRI	LOW
7	OUTPUT	Request To Send	RTS	LOW
8	INPUT	Clear To Send	CTS	LOW
9	INPUT	Ring Indicator	RI	LOW

DCE – Female DB9 Socket

Pin Number	I/O	Name	Abbreviation	Active
1	OUTPUT	Carrier Detect	CD	LOW
2	OUTPUT	Transmitted Data Out	TDO	
3	INPUT	Received Data In	RDI	
4	INPUT	Data Terminal Ready Input	DTRI	LOW
5	GROUND	Signal Ground	GND	n/a
6	OUTPUT	Data Set Ready	DSR	LOW
7	INPUT	Request To Send	RTS	LOW
8	OUTPUT	Clear To Send	CTS	LOW
9	OUTPUT	Ring Indicator	RI	LOW

Bi-directional Data link.

Now to the business end of this discussion. Forget about the telephone line and the Modems, we need to create a bi-directional serial RS232C link between the controlling computer and our Machine Controller. To do this we need to understand the communication hardware provided by each unit. Firstly the controlling computer is an x86 PC running the Linux Operating System with two serial ports which we will call 'COM1' and 'COM2'. Both of the PC's serial ports are configured as DTE which means on a modern PC that they are male DB9 (9 pin) sockets. The Machine Controller has also got two serial ports which we will call 'serial0' and 'serial1'. 'serial1' is configured as DTE and 'serial0' as DCE. Points to be considered are,

- The ports on the Machine Controller will need to be a female DB9 outlet socket for serial0 and a male DB9 outlet socket for serial1.
- We will be communicating between the two devices at 9600 Baud, one start bit and two stop bits with no Parity checking using 7 bit ASCII code.
- IDC (Insulation Displacement Connection) cables need to be made which will connect the 10 pin male headers on the Machine Controller to the DB9 outlet sockets.
- A communication cable will be needed to connect the Machine Controller DCE port 'serial0' to one of the DTE ports on the PC (usually COM2).
- Software will be required on the PC to take the raw 'Instruction Code' or 'Monitor' commands generated and transfer them to/from the Machine Controller.
- There needs to be a 'Monitor' program on board the Machine Controller to respond to 'Machine Instructions' or 'Monitor Commands' sent from the PC.
- There needs to be diagnostic routines on board the Machine Controller to read/alter register and memory locations.
- There needs to be a handshake line from the Machine Controller to the controlling Linux PC to indicate that the Machine Controller is ready to receive data or commands.
- There needs to be a DTE port available on the Machine Controller ('serial1') to communicate with either a machine to be controlled or another PC for operational and diagnostic purposes.
- There needs to be a handshake line from either a machine to be controlled or another PC for operational and diagnostic purposes to the Machine Controller to indicate to the Machine Controller when the machine or other PC is ready to receive instructions.

The ten pin male headers of the Machine Controller's two serial ports ('serial0' and 'serial1') are connected to their male and female sockets respectively via Insulation Displacement Ribbon Cables. IDC fittings are used with the recommended flat ribbon cable and are connected according to the following pin out table,

10 Pin Headers viewed from top			***IDC DB9 Socket pin out***	
Pin Number	SERIAL0 (DCE)	SERIAL1 (DTE)	Pin Number	RS232C Signal Name
1	n/c	n/c	1	no connect
2	TX1	RX2	2	TD/RD
3	RX1	TX2	3	RD/TD
4	DTR-IN	DTR-OUT	4	DTR
5	GND	GND	5	Signal Ground
6	n/u	n/u	not used	not used
7	n/c	n/c	9	no connect
8	n/c	n/c	8	no connect
9	n/c	n/c	7	no connect
10	DSR-OUT	DSR-IN	6	DSR