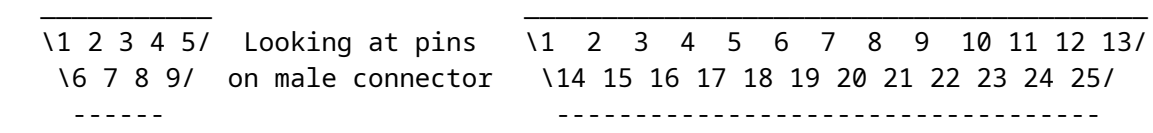
Next Previous Contents

19. Pinout and Signals

19.1 Pinout of 9-pin and 25-pin serial connectors

The pin numbers are often engraved in the plastic of the connector but you may need a magnifying glass to read them. Note DCD is sometimes labeled CD. The numbering of the pins on a female connector is read from right to left, starting with 1 in the upper right corner (instead of 1 in the upper left corner for the male connector as shown below). --> direction is out of PC.



Pin # 9-pin	Pin # 25-pin	Acronym	Full-Name	Direction	What-it-May-Do/Mean
3	2	TxD	Transmit Data	-->	Transmits bytes out of PC
2	3	RxD	Receive Data	<--	Receives bytes into PC
7	4	RTS	Request To Send	-->	RTS/CTS flow control
8	5	CTS	Clear To Send	<--	RTS/CTS flow control
6	6	DSR	Data Set Ready	<--	I'm ready to communicate
4	20	DTR	Data Terminal Ready	-->	I'm ready to communicate
1	8	DCD	Data Carrier Detect	<--	Modem connected to another
9	22	RI	Ring Indicator	<--	Telephone line ringing
5	7	SG	Signal Ground		

9-Pin DB9 Connector			25-Pin DB-25 Connector		
1	DCD	Carrier Detect	1		Chassis Ground
2	RxD	Receive Data	2	TxD	Transmit Data
3	TxD	Transmit Data	3	RxD	Receive Data
4	DTR	Data Terminal Ready	4	RTS	Request To Send
5	SG	Signal Ground	5	CTS	Clear To Send
6	DSR	Data Set Ready	6	DSR	Data Set Ready
7	RTS	Request To Send	7	SG	Signal Ground
8	CTS	Clear To Send	8	DCD	Carrier Detect
9	RI	Ring Indicator	20	DTR	Data Terminal Ready
			22	RI	Ring Indicator

19.2 Signals May Have No Fixed Meaning

Only 3 of the 9 pins have a fixed assignment: transmit, receive and signal ground. This is fixed by the hardware and you can't change it. But the other signal lines are controlled by software and may do (and mean) almost anything at all. However they can only be in one of two states: asserted (+12 volts) or negated (-12 volts). Asserted is "on" and negated is "off". For example, Linux software may command that DTR be negated and the hardware only carries out this command and puts -12 volts on the DTR pin. A modem (or other device) that receives this DTR signal may do various things. If a modem has been configured a certain way it will hang up the telephone line when DTR is negated. In other cases it may ignore this signal or do something else when DTR is negated (turned off).

It's like this for all the 6 signal lines. The hardware only sends and receives the signals, but what action (if any) they perform is up to the Linux software and the configuration/design of devices that you connect to the serial port. However, most pins have certain functions which they normally perform but this may vary with the operating system and the device driver configuration. Under Linux, one may modify the source code to make these signal lines behave differently (some people have).

19.3 Cabling Between Serial Ports

A cable from a serial port always connects to another serial port. An external modem or other device that connects to the serial port has a serial port built into it. For modems, the cable is always straight thru: pin 2 goes to pin 2, etc. The modem is said to be DCE (Data Communications Equipment) and the computer is said to be DTE (Data Terminal Equipment). Thus for connecting DTE-to-DCE you use straight-thru cable. For connecting DTE-to-DTE you must use a null-modem cable (also called a crossover cable). There are many ways to wire such cable (see examples in Text-Terminal-HOWTO subsection: "Direct Cable Connection")

There are good reasons why it works this way. One reason is that the signals are unidirectional. If pin 2 sends a signal out of it (but is unable to receive any signal) then obviously you can't connect it to pin 2 of the same type of device. If you did, they would both send out signals on the same wire to each other but neither would be able to receive any signal. There are two ways to deal with this situation. One way is to have a two different types of equipment where pin 2 of the first type sends the signal to pin 2 of the second type (which receives the signal). That's the way it's done when you connect a PC (DTE) to a modem (DCE). There's a second way to do this without having two different types of equipment: Connect pin sending pin 2 to a receiving pin 3 on same type of equipment. That's the way it's done when you connect 2 PCs together or a PC to a terminal (DTE-to-DTE). The cable used for this is called a null-modem cable since it connects two PCs without use of a modem. A null-modem cable may also be called a cross-over cable since the wires between pins 2 and 3 cross over each other (if you draw them on a sheet of paper). The above example is for a 25 pin connector but for a 9-pin connector the pin numbers 2 and 3 are just the opposite.

The serial pin designations were originally intended for connecting a dumb terminal to a modem. The terminal was DTE (Data Terminal Equipment) and the modem was DCE (Data Communication Equipment). Today the PC is usually used as DTE instead of a terminal (but real terminals may still be used this way). The names of the pins are the same on both DTE and DCE. The words: "receive" and "transmit" are from the "point of view" of the PC (DTE). The transmit pin from the PC transmits to the "transmit" pin of the modem (but actually the modem is receiving the data from this pin so from the point of view of the modem it would be a receive pin).

The serial port was originally intended to be used for connecting DTE to DCE which makes cabling simple: just use a straight-thru cable. Thus when one connects a modem one seldom needs to worry about which pin is which. But people wanted to connect DTE to DTE (for example a computer to a terminal) and various ways were found to do this by fabricating various types of special null-modem cables. In this case what pin connects to what pin becomes significant.

19.4 RTS/CTS and DTR/DSR Flow Control

This is "hardware" flow control. Flow control was previously explained in the Flow Control subsection but the pins and voltage signals were not. Linux only supports RTS/CTS flow control at present (but a special driver may exist for a specific application which supports DTR/DSR flow control). Only RTS/CTS flow control will be discussed since DTR/DSR flow control works the same way. To get RTS/CTS flow control one needs to either select hardware flow control in an application program or use the command:
stty -F /dev/ttyS2 crtscts (or the like). This enables RTS/CTS hardware flow control in the Linux device driver.

Then when a DTE (such as a PC) wants to stop the flow into it, it negates RTS. Negated "Request To Send" (-12 volts) means "request NOT to send to me" (stop sending). When the PC is ready for more bytes it asserts RTS (+12 volts) and the flow of bytes to it resumes. Flow control signals are always sent in a direction opposite to the

flow of bytes that is being controlled. DCE equipment (modems) works the same way but sends the stop signal out the CTS pin. Thus it's RTS/CTS flow control using 2 lines.

On what pins is this stop signal received? That depends on whether we have a DCE-DTE connection or a DTE-DTE connection. For DCE-DTE it's a straight-thru connection so obviously the signal is received on a pin with the same name as the pin it's sent out from. It's RTS-->RTS (PC to modem) and CTS<--CTS (modem to PC). For DTE-to-DTE the connection is also easy to figure out. The RTS pin always sends and the CTS pin always receives. Assume that we connect two PCs (PC1 and PC2) together via their serial ports. Then it's RTS(PC1)-->CTS(PC2) and CTS(PC1)<--RTS(PC2). In other words RTS and CTS cross over. Such a cable (with other signals crossed over as well) is called a "null modem" cable. See Cabling Between Serial Ports

What is sometimes confusing is that there is the original use of RTS where it means about the opposite of the previous explanation above. This original meaning is: I Request To Send to you. This request was intended to be sent from a terminal (or computer) to a modem which, if it decided to grant the request, would send back an asserted CTS from its CTS pin to the CTS pin of the computer: You are Cleared To Send to me. Note that in contrast to the modern RTS/CTS bi-directional flow control, this only protects the flow in one direction: from the computer (or terminal) to the modem. This original use appears to be little used today on modern equipment (including modems).

The DTR and DSR Pins

Just like RTS and CTS, these pins are paired. For DTE-to-DTE connections they are likely to cross over. There are two ways to use these pins. One way is to use them as a substitute for RTS/CTS flow control. The DTR pin is just like the RTS pin while the DSR pin behaves like the CTS pin. Although Linux doesn't support DTR/DSR flow control, it can be obtained by connecting the RTS/CTS pins at the PC to the DSR/DTR pins at the device that uses DTR/DSR flow control. DTR flow control is the same as DTR/DSR flow control but it's only one-way and only uses the DTR pin at the device. Many text terminals and some printers use DTR/DSR (or just DTR) flow control. In the future, Linux may support DTR/DSR flow control. The software has already been written but it's not clear when (or if) it will incorporated into the serial driver.

The normal use of DTR and DSR (not for flow control) is as follows: A device asserting DTR says that its powered on and ready to operate. For a modem, the meaning of a DTR signal from the PC depends on how the modem is configured. Negating DTR is sometimes called "hanging up" but it doesn't always do this. One way to "hang up" (negate DTR) is to set the baud rate to 0 using the command "stty 0". Trying to do this from a "foreign" terminal may not work due to the two-interface problem. See Two interfaces at a terminal. For internal modem-serial_ports it worked OK with a port using minicom but didn't work if the port was using wvdial. Why?

19.5 Preventing a Port From Opening

If "stty -clocal" (or getty is used with the "local" flag negated) then a serial port can't open until DCD gets an assert (+12 volts) signal.

Next Previous Contents