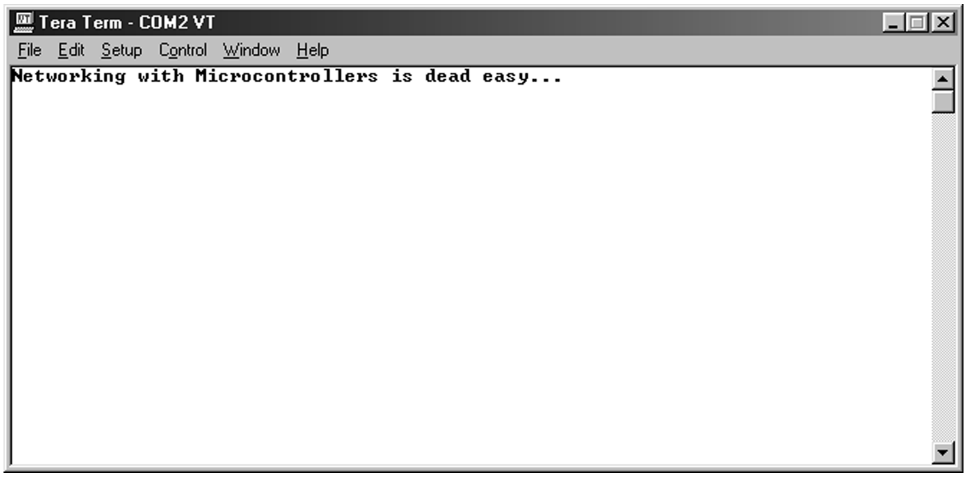
CHAPTER 1

The Essence of Microcontroller Networking—RS-232

Let's begin by exploring the RS-232 protocol. Knowing how to manipulate data with RS-232 will help you master more complex communications protocols. You'll also find RS-232 techniques to be invaluable in the development phase of your projects.



```
Networking with Microcontrollers is dead easy...
```

Figure 1.1: Effecting RS-232 communications with a microcontroller is a snap. As you continue reading this book, you will find that knowing how to implement simple RS-232 with a microcontroller can assist you in building and debugging more complex microcontroller projects.

The information you see in the terminal emulator window in Figure 1.1 was generated by some very simple firmware and a not-so-complicated off-the-shelf, two-buck microcontroller. I used a tiny 8-bit microcontroller that does not contain a built-in hardware USART (Universal Synchronous/Asynchronous Receiver/Transmitter), to transfer the ASCII characters you see in Figure 1.1 from one of its I/O pins to an RS-232 converter IC. A serial cable connected between the microcontroller/RS-232 converter IC circuitry and my personal computer's serial port allowed the ASCII characters to flow from the little microcontroller's firmware out of the microcontroller's I/O pin, through the RS-232 converter IC, across the serial cable to the personal computer's USART/RS-232 circuitry and finally end up in the terminal emulator window you see in Figure 1.1.

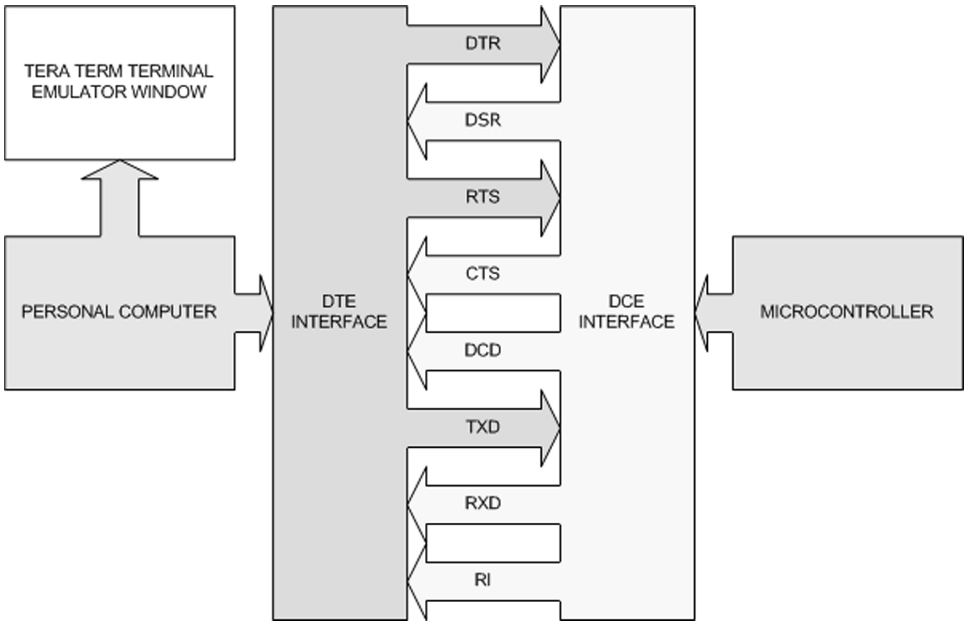


Figure 1.2: The DTE and DCE interfaces usually consist of some sort of voltage-conversion circuitry to translate RS-232 voltage levels to voltage levels that are compatible with the computing equipment on each end of the communications link. The simplest form of an RS-232 link uses only the TXD and RXD signals with a common ground.

What I've just described is one of the simplest forms of microcontroller networking. It is commonly known as serial or RS-232 communications. As you can see in Figure 1.2, RS-232 was designed to tie DTE (Data Terminal Equipment) and DCE (Data Communications Equipment) devices together electronically to effect bidirectional data communications between the devices.

An example of a DTE device is the serial port on your personal computer. Under normal conditions, the DTE interface on your personal computer asserts DTR (Data Terminal Ready) and RTS (Request To Send). DTR and RTS are called modem control signals. A typical DCE device interface responds to the assertion of DTR by activating a signal called DSR (Data Set Ready). The DTE RTS signal is answered by CTS (Clear To Send) from the DCE device. A standard external modem that you would connect to your personal computer serial port is a perfect example of a DCE device.

Some History

In May of 1960, it was evident that a standard was needed to identify the electrical interface between computers and modems. It was decided to establish a standard voltage with standard signal parameters and a standard nomenclature to identify the conductors in the cable that connected computers and data sets. Even today, you will sometimes hear the term data set applied to modems and DCE equipment.

To compete as well as exist in the current communications environment, telecommunications vendors needed common ground to assure that each vendor's equipment set could talk to any other vendor's telecommunications equipment set. In other words, the industry needed a working standard. Without a standard, the whole teleprocessing industry could come to a grinding, nonstandardized halt.

To help establish some harmony, a committee named the Electronic Industries Association was formed. The EIA drafted a standard known as EIA RS-232(X). Though it was a great idea, the original specification was broad in meaning and didn't guarantee compatibility. The new RS-232 specification also had a competitor outside the United States, known as the CCITT, or Consultative Committee on International Telegraphy and Telephony, recommendation V.24.

The RS-232 proposal defined a logical and physical interface between DTE equipment and DCE equipment. The computer's DTE serial port presents both a physical and a logical interface to a modem or data set's DCE port and consists of several conductors for controlling, transmitting and receiving data. Timing and clocking signals are also intermixed within the RS-232 interface. The logical and physical attributes of the RS-232 proposal eventually became a set of standards known today as the EIA RS-232 interface.

Once the signals reach the DCE device, a second interface provides a physical path to the communication channel (RF link, telephone line, fiber-optic link, satellite link, and so forth). For most of you, that second interface is a standard two-conductor analog telephone line, which is terminated inside your modem.

The EIA standard originally identified seven interface conductors and no specific connector. Signal voltages were defined as at least 3 volts but not greater than 20 volts with respect to ground.

In October 1963, RS-232 became RS-232-A and was modified to include a 25-pin connector with a maximum cable length of 50 feet. This revision established fixed relationships between a circuit and specific pin numbers on the 25-pin connector. Also, an alphabetic coding system for each type of interface circuit was presented. The first character of the coding system designated A for ground, B for data, C for control and D for clocking. Table 1.1 lays out the pinout and various names for each RS-232 signal.

Pin	Line Label	Line Name	Signal Direction	Level
1	AA	Positive Ground	N.A.	A,B C
2	BA	Transmitted Data	To DCE	A B,C
3	BB	Received Data	To DTE	A,B,C
4	CA	Request To Send	To DCE	A B,C
5	CB	ClearTo Send	To DTE	A B,C
6	CC	Data Set Ready	To DTE	A B,C
7	AB	Signal Ground	N.A.	A B C
8	CF	Received Line Signal Detector (RS-232); Data Carrier Detect (RS-232A/B)	To DTE	A,B,C
11	N.A.	Select Standby	To DCE	C
12	SCF	Secondary Receive Line Signal Detector	To DTE	C
13	SCB	Secondary Clear To Send	To DTE	C
14	SBA	Secondary Transmitted Data	To DCE	C
14	N.A.	New Sync	To DCE	A,B,C
15	DB	Transmitter Signal Element Timing	To DTE	A B C
16	SBB	Secondary Received Data	To DTE	C
17	DD	Receiver Signal Element Timing	To DTE	A,B,C
18	N.A.	Test	To DCE	C
19	SCA	Secondary Request To Send	To DCE	C
20	CD	Data Terminal Read	To DCE	A,B,C
21	CG	Signal Quality Detector	To DTE	C
22	CE	Indicate Ring/Calling	To DTE	A,B,C

Table 1.1: Specifications list for RS-232 interface.

There are a couple of confusion points. Note the total lack of logic when associating DB-25 pins with DB-9 pins. And, this table is based on the DTE side of the circuit. To get things to work, you must switch the TD and RD pins on the DCE side of the circuit. When you do the switch that puts the DTE TD pin's data into the DCE RD pin and the DCE's TD pin's data into the DTE RD pin. If you're using the modem signals, you have to tie them together properly between the DTE and DCE as well.

The original seven basic circuits and the signal-level definition of –3 volts for mark and +3 volts for space were retained intact, adding ten additional optional circuit definitions. The maximum permissible open-circuit voltage was changed to 25 volts, and a current maximum between any two conductors, including ground, was set at 0.5 ampere. Conductors that permit auto-answer capability were first introduced in this revision.

October 1965 brought about RS232-B, which defined terminating impedances that permitted circuit designers to build hardware with greater reliability. Open-circuit signal levels remained unchanged at –3 to –25 volts as mark and +3 to +25 volts as space, but revision B added an important voltage specification. By specifying that signal ground on pin 7 be tied to frame ground on pin 1 in the DCE equipment, a definite signal reference is established between DTE and DCE devices.

The *Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange* specification was released in August 1969. It further clarified conductor definitions and stated that properly terminated RS-232 circuits shall not exceed ±15 volts.

RS-232-C came along later and defined the interface between Data Terminal Equipment (DTE) and Data Circuit terminating Equipment (DCE). In the early days, a piece of DTE hardware was usually a dumb terminal. DEC's (Digital Equipment Corporation in those days; Hewlett-Packard/COMPAQ these days) VT100 was and is the most well-known dumb terminal and is still emulated today.

As you would imagine, a standard DTE device should be capable of emitting and receiving a serial data stream. As you have already seen, that includes microcontrollers and personal computers in the "could be a DTE" category. Although DCE equipment can also transmit and receive a serial data stream, the primary purpose of DCE equipment is to receive the DTE-generated bit stream over an RS-232 interface and convert it to a form that's suitable for transmission over a telecommunication medium. In the case of a personal computer modem, that telecommunications medium is most likely a voice-grade telephone line.

Ever noticed that every serial port interface on your personal computer is male and every modem serial port interface you've ever seen is female? There's a reason for that. The RS-232-C standard states that physical DTE port connectors will be male and physical DCE port connectors will be female.

Older personal computers and modems used a 25-pin connector. Today's 9-pin serial connectors aren't really standards although they have become so by proxy. The 9-pin interface first appeared commercially on AT-class PCs in the early 1980s.

RS-232 Standard Operating Procedure

Today, the majority of commercially available equipment is based on the RS-232-C or RS-232-D standard. (The CCITT V.24 and V.28 standards are also common and widely-used.) There are 25 circuits defined in the RS-232 standard. The good news is that most of the 25 RS-232 circuits don't have to be used to effect an asynchronous communications session between a DTE and DCE device. Things could be different for synchronous communications sessions that employ complex communications protocols and that's why the timing and clocking signals are defined in the RS-232 standard. There's a good reason that a 9-pin connector is on your personal computer instead of the standard appointed 25-pin connector. You only need nine RS-232 signal lines to communicate asynchronously using a standard asynchronous modem. Let's look at them from a "commented" standards point of view.

- *Pin 1 (Protective Ground Circuit, AA).* This conductor is bonded to the equipment frame and can be connected to external grounds if other regulations or applications require it.

  Comment: Normally, this is either left open or connected to the signal ground. This signal is not found in the DTE 9-pin serial connector.

- *Pin 2 (Transmitted Data Circuit BA, TD).* This is the data signal generated by the DTE. The serial bit stream from this pin is the data that's ultimately processed by a DCE device.

  Comment: This is pin 3 on the DTE 9-pin serial connector. This is one of the three minimum signals required to effect an RS-232 asynchronous communications session.

- *Pin 3 (Received Data Circuit BB, RD).* Signals on this circuit are generated by the DCE. The serial bit stream originates at a remote DTE device and is a product of the receive circuitry of the local DCE device. This is usually digital data that's produced by an intelligent DCE or modem demodulator circuitry.

  Comment: This is pin 2 on the DTE 9-pin serial connector. This is another of the three minimum signals required to effect an RS-232 asynchronous communications session.

- *Pin 4 (Request To Send Circuit CA, RTS).* This signal prepares the DCE device for a transmit operation. The RTS ON condition puts the DCE in transmit mode, while the OFF condition places the DCE in receive mode. The DCE should respond to an RTS ON by turning ON Clear to Send (CTS). Once RTS is turned OFF, it shouldn't be turned ON again until CTS has been turned OFF. This signal is used in conjunction with DTR, DSR and DCD. RTS is used extensively in flow control.

  Comment: This is pin 7 on the DTE 9-pin serial connector. In simple 3-wire implementations this signal is left disconnected. Sometimes you will see this signal tied to the CTS signal to satisfy a need for RTS and CTS to be active signals in the communications session. You will also see RTS feed CTS in a null modem arrangement.

- *Pin 5 (Clear To Send Circuit CB, CTS).* This signal acknowledges the DTE when RTS has been sensed by the DCE device and usually signals the DTE that the DCE is ready to accept data to be transmitted. Data is transmitted across the communications medium only when this signal is active. This signal is used in conjunction with DTR, DSR and DCD. CTS is used in conjunction with RTS for flow control.

  Comment: This is pin 8 on the DTE 9-pin serial connector. In simple 3-wire implementations this signal is left disconnected. Otherwise, you'll see it tied to RTS in null modem arrangements or where CTS has to be an active participant in the communications session.

- *Pin 6 (Data Set Ready Circuit CC, DSR).* DSR indicates to the DTE device that the DCE equipment is connected to a valid communication medium and, in some cases, indicates that the line is in the OFF HOOK condition. OFF HOOK is an indication that the DCE is either in dialing mode or in session with another remote DCE. When this signal is OFF, the DTE should be instructed to ignore all other DCE signals. If this signal is turned off before DTR, the DTE is to assume an aborted communication session.

Comment: This is pin 6 on the DTE 9-pin serial connector. DSR is sometimes used in a flow control arrangement with DTR. Some modems assert DSR when power to the modem is applied regardless of the condition of the communications medium.

- *Pin 7 (Signal Common Circuit, AB).* This conductor establishes the common-ground reference for all interchange circuits, except Circuit AA, protective ground. The RS-232-B specification permits this circuit to be optionally connected to protective ground within the DCE device as necessary.

  Comment: This is pin 5 on the DTE 9-pin serial connector and is the only ground connection. This is the third wire of the minimal 3-wire configuration. Thus, an RS-232 asynchronous communications session can be effected with only three signals: TX (Transmit Data), RX (Receive Data) and signal ground.

- *Pin 8 (Data Carrier Detect Circuit CF, DCD).* This pin is also known as Received Line Signal Detect (RSLD) or Carrier Detect (CD). This signal is active when a suitable carrier is established between the local and remote DCE devices. When this signal is OFF, RD should be clamped to the mark state (binary 1).

  Comment: This is pin 1 on the DTE 9-pin serial connector. Normally in use only if a modem is in the communications signal path. You will also see this signal tied active in a null modem arrangement.

- *Pin 20 (Data Terminal Ready Circuit CD, DTR).* DTR signals are used to control switching of the DCE to the communication medium. DTR ON indicates to the DCE that connections in progress shall remain in progress, and if no sessions are in progress, new connections can be made. DTR is normally turned off to initiate ON HOOK (hang-up) conditions. The normal DCE response to activating DTR is to activate DSR.

  Comment: This is pin 4 on the DTE 9-pin serial connector. Unless you specify differently or run a program that controls DTR, usually it is present on the personal computer serial port as long as the personal computer is powered on. Occasionally you will see this signal used in flow control.

- *Pin 22 (Ring Indicator Circuit CE, RI).* The ON condition of this signal indicates that a ring signal is being received from the communication medium (telephone line). It's normally up to the control program to act on the presence of this signal.

  Comment: This is pin 9 on the DTE 9-pin serial connector. This signal follows the incoming ring to an extent. Normally, this signal is used by DCE auto-answer algorithms.

That is all that's needed RS-232 signal-wise to establish a session between a DTE and a DCE device. Now that you have a feeling for what each RS-232 signal does, let's review how they react to each other with respect to the transfer of data between a DTE and DCE device.

- Local DTE (personal computer, microcontroller, etc.) is powered up and DTR is asserted.
- Local DCE (modem, data set, microcontroller, etc.) is powered up and senses the DTR from the local DTE.
- Local DCE asserts DSR. If the DCE device is a modem, it goes off-hook (picks up the line). If a dial-up session is to be established, the DTE sends a dial instruction and phone number to the modem.
- If the line is good and the other end (remote DCE) is ready or answers the dial-up from the local DCE, a carrier is generated/detected and the local and remote DCE devices assert DCD. The session is established.
- The transmitting DTE raises RTS.
- The transmitting DCE responds with CTS.
- The control program transmits or receives data.

In our historical review, the DTE or personal computer and DCE or modem took care of converting the RS-232 signal levels to appropriate personal computer circuitry levels. To perform RS-232 asynchronous communications with microcontrollers, we must employ a voltage translation scheme of our own. Fortunately, there are many ways to do this and all of them are easy to implement.

RS-232 Voltage Conversion Considerations

RS-232 converter ICs like those made by Maxim and Sipex convert the negative RS-232 voltages to positive logic voltage levels that microcontroller circuits can understand. The positive RS-232 voltages are converted to a microcontroller's logical 0 (zero) voltage level. If the microcontroller circuitry is powered by +5 VDC, then an RS-232 '1' or mark is converted to a TTL (Transistor Transistor Logic) *high* or '1' and an RS-232 '0' or space is translated into a TTL *low* or '0'. With the advent of 3-volt logic, special RS-232 converter ICs that can operate at the 3-volt power supply levels have been introduced. The bottom line is that the RS-232 marks and spaces must be converted to voltage levels the microcontroller can understand before any communications and data transfer can be realized between devices.

In reality, the full-positive and negative voltage swing called out by the RS-232 standard doesn't have to be employed to effect RS-232 communications links. With the right cable an RS-232 voltage of –3 volts is sufficient to generate a '1' or mark while +3 volts will produce a '0' or space. The area between –3 volts and +3 volts (shown in Figure 1.3) is a transition zone and is where most of the nasty line noise can and should be found. By defining this ±3-volt threshold, the signal-to-noise ratio of the RS-232 physical link is improved. If a high-quality serial cable is used and the distance between stations is relatively short, RS-232 voltages that resemble microcontroller logic voltages can be used to transfer information

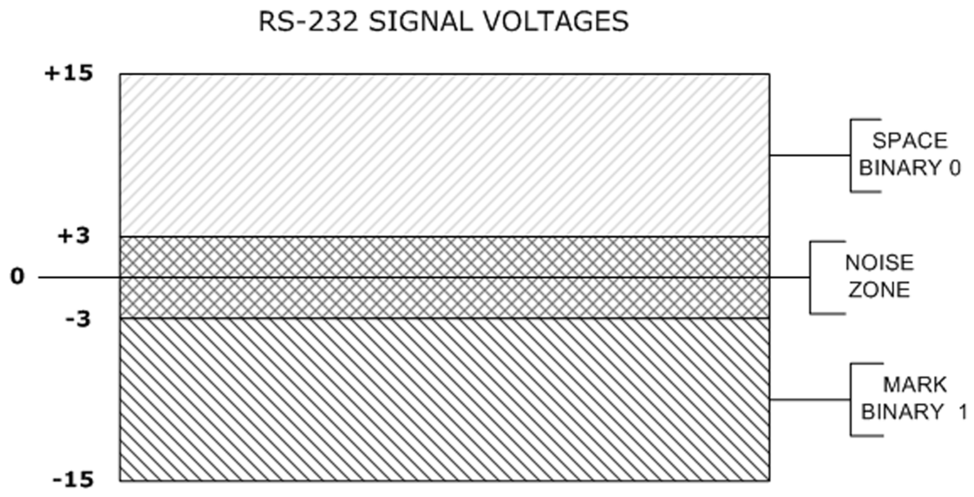


Figure 1.3: Cheap RS-232 implementations dare to use the 0 VDC to +5 VDC region for marks and spaces with 0 VDC being a mark and anything over +3 VDC representing a space. The "NOISE ZONE" I've marked is actually called the transition zone.

between a DTE and DCE device. In addition, using a high-quality cable could extend the 50-foot maximum cable length specified by the RS-232 specification. Reducing the speed of the data transmission can also extend the maximum cable length between a wired set of DTE and DCE devices as well.

The good news is that you don't have to know the nitty-gritty details of the RS-232 specification to use RS-232 as a means of communicating with a microcontroller. In fact, I've already given you more RS-232 history and theory than you really need to know to make a microcontroller talk asynchronously. In this book, we're all about the practical application of RS-232 as it pertains to microcontrollers. So, let's look at some RS-232 hardware and the firmware behind it.