chapter 1

X-10 Under the Hood

One of my advisors will be an average five-year-old child. Any flaws in my plan that he is able to spot will be corrected before implementation.

Peter Anspach,
The Evil Overlord List

in this chapter

- ☑ Power line control
- ☑ Layered communications model
- ☑ X-10 power line protocols
- ☑ Standard modules
- ☑ Troubleshooting and noise
- ☑ Installation and wiring
- ☑ Custom modules

Home automation has much in common with car modification and repair — anyone can plug in parts or bolt on wheels, but you need to look under the hood and understand what's going on in there before you take on anything significant. This chapter will teach you the internals of how X-10 home automation works and give you the know-how to design and build your own systems that work reliably and do what you want.

Before you read on, please read this caution: *Power line control devices, such as the ones we cover in this chapter, operate on hazardous power line voltages. Touching the wrong wire at the wrong time can kill you. If you don't know how to work on home power wiring safely, stick to the modules that plug in and don't attempt anything more adventurous without trained help.*

Project Description

X-10 home automation systems transmit signals over your house wiring, a difficult challenge because of noise and transmission problems inherent in using the wiring for an application it's not designed to handle. Building successful X-10 systems requires that you

- ➤ Understand how the technology works so you know what's possible within the framework and constraints of the technology
- ➤ Know what components you have to work with so you design low-cost systems you can readily implement
- ➤ Anticipate the problems your system will encounter and take steps to avoid or cure them

We'll start by looking at the technology as a network, covering how signals operate on the wires and how the system transports messages using those signals. We'll then look at what can go wrong and what those problems do in your system. Along the way, you'll learn about protocols, and software, noise, and signal measurements.

System Diagram

Figure 1-1 diagrams the overall system for the simplest X-10 application, remote control of a device. Instead of the typical circuit in which the switch controls the power flowing to the lamp, the switch signals an X-10 transmitter. That transmitter sends a message to the X-10 receiver using signals over the power lines. The receiver detects the signals, decodes the message, and controls the power flow to the lamp according to the content of the message. Messages can command the lamp to be on, off, or dimmed to a specified level.

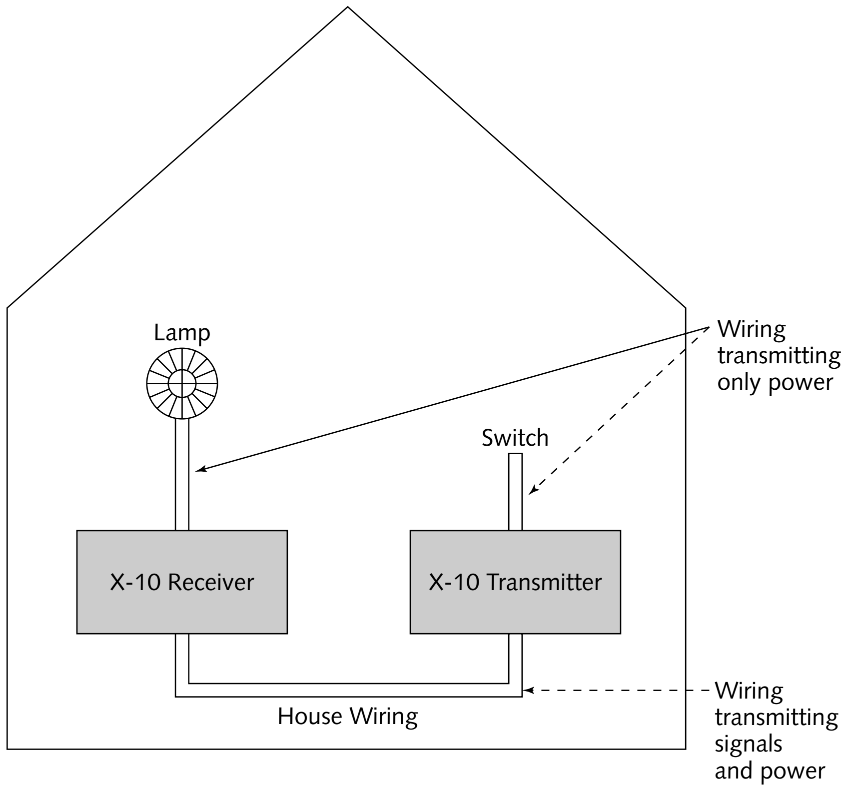

FIGURE 1-1: Simplest X-10 system diagram

The standard ISO seven-layer communications model (Figure 1-2) describes any communication system, and so applies to X-10 systems. We'll start with the physical layer and work upwards. X-10 systems don't implement layers three (networking) through five (session), so we'll skip directly from the link layer to the presentation layer.

7: Application	Specifies interactions among applications
6: Presentation	Controls information formatting for display or print, and data encryption
5: Session	Monitors and brokers communication between systems, including security, logging, and administrative functions
4: Transport	Supports end-to-end movement of data between systems, and ensures error-free transmissions through error checking and correction or retransmission
3: Network	Routes data between systems across the network to ensure data arrives at the correct destination
2: Data Link	Defines the rules for sending and receiving information from one computer to another
1: Physical	Implements the physical transfer of information between computers

FIGURE 1-2: ISO seven-layer model

Our discussion starts with the physical layer, showing how information physically moves from one device to another.

Physical layer

The physical layer in an X-10 system transmits data over the power lines. The challenge is to send the signal reliably, with relatively low-power, inexpensive electronics. X-10 chose a simple scheme using a tone or its absence to convey a one or zero data bit, respectively, and uses redundant signaling to help improve reliability.

Figure 1-3 shows the fundamental approach. An X-10 transmitter will output a 120 kHz tone in a 1 ms burst starting when a rising line voltage crosses the zero point. The presence of the tone signals a one bit; the absence of a tone signals a zero. The burst is further transmitted two more times, at 1.047 radian intervals (2π / 6 radians, dividing the half cycle in thirds), to provide signals that will have appropriate zero crossing timing when bridged to the alternate phases in a three-phase circuit.

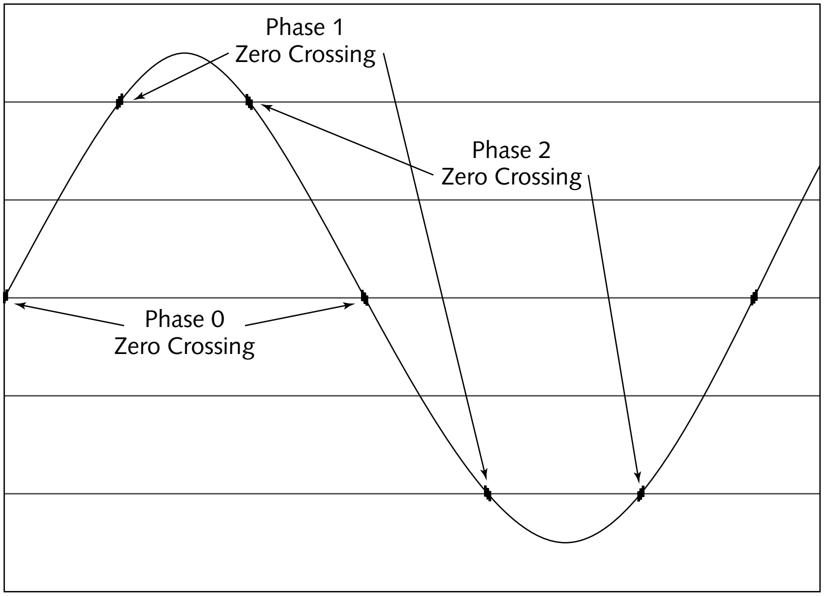

FIGURE 1-3: X-10 signal transmission

Figure 1-3 shows bursts at the rising voltage zero crossing points, along with bursts starting at the zero crossing for the negative-going power line voltage. The X-10 signal protocol sends the complement of the bit being transmitted at that time (and in the following bursts for the other line phases) to provide error checking at the receiver on the received signal. For example, if there is a tone at the rising voltage zero crossing, representing a one bit, there will not be any tone at the falling voltage zero crossing, and vice versa. This approach provides no error correction, but helps the receiver detect when a bit has been corrupted.

Timing the signal at the zero crossings simplifies the receivers, reducing cost, because the underlying 60 Hz carrier provides precise timing for the window in which the 120 kHz tone will arrive. The physical layer protocol requires that the tone burst start no more than 200 µ (microseconds) after the zero crossing, defining the limits within which a tone detector must operate.

Because the system only transmits one bit per cycle of the underlying carrier, the raw signaling bit rate of the X-10 system is 60 bits per second (bps). The actual payload data rate is less, as shown in the next section. Because of the very low data rate, and because the physical layer has no provision to detect or avoid collisions, it's important that X-10 messages be short and infrequent.

Link layer

All the physical layer does is convey bits over the power line. The link layer is responsible for conveying information from one device to another, including providing mechanisms for framing messages and rules for sending and receiving messages.

The first task the link layer must accomplish is to mark the start of each message so the receiver can reliably know where in a bit stream the actual message starts. X-10 does this with a unique start code (Figure 1-4), a signal pattern that cannot occur within a message. That start code is the sequence tone-tone-tone-no tone (1110) transmitted on sequential half cycles, starting at a rising voltage zero crossing. The first two zero crossings — the first full waveform cycle — are an invalid sequence, violating the requirement that the complement of a rising voltage zero crossing be transmitted on the following zero crossing. The second full cycle is a valid sequence, representing a one bit. Including an invalid sequence in the start code, an instance of what is more generally called *out-of-band signaling*, ensures that no error-free message can be mistaken for a start code.

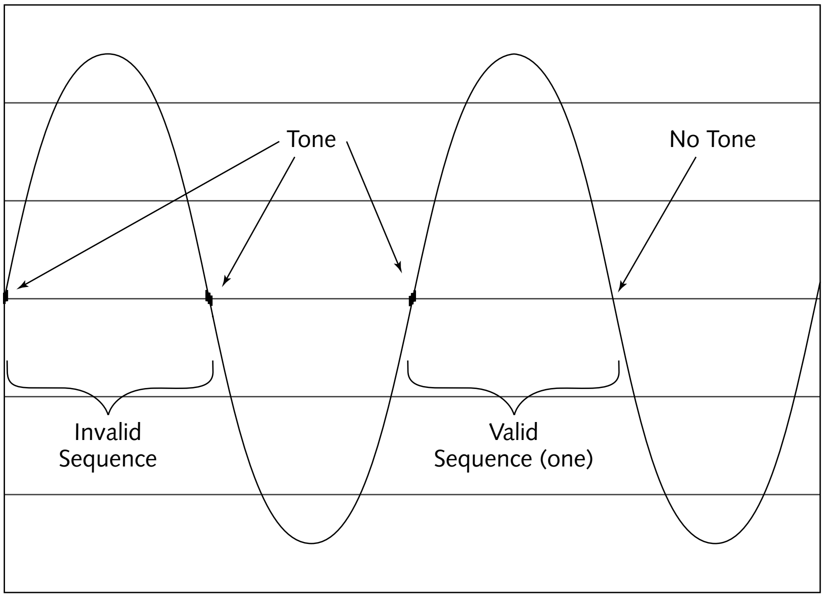

FIGURE 1-4: X-10 start code

Nine bits of data immediately follow the start code in the subsequent nine cycles, as shown in Figure 1-5. The information in the message is comprised of four bits of *house code*, followed by five bits of *key number* or *function code*. The house code is simply a group address, while the key number identifies the specific device within the group. Function codes are operations such as *on* or *off*.

Start Code	Cycle	Tone Tone
	Cycle	Tone No Tone
House Code	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
Key Code	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse
	Cycle	Bit Bit Inverse

FIGURE 1-5: X-10 standard message format

In another layer of error detection, every message gets transmitted twice. Ordinary messages should have three full empty cycles between messages, but bright/dim commands should be sequential, with no intervening empty cycles.

Presentation and application layers

The presentation and application layers define the content of the messages and how they're interpreted. House codes are represented by specific sequential four-bit patterns, as shown in Table 1-1. Bit 0 is transmitted first, followed by the rest in order. The bit patterns for the codes do not correspond to ASCII codes, which would represent "A" as 1000001, but are unique to the X-10 system.

Table 1-1 X-10 House Code Bit Patterns

Code	*Bit 0*	*Bit 1*	*Bit 2*	*Bit 3*
A	0	1	1	0
B	1	1	1	0
C	0	0	1	0
D	1	0	1	0
E	0	0	0	1
F	1	0	0	1
G	0	1	0	1
H	1	1	0	1
I	0	1	1	1
J	1	1	1	1
K	0	0	1	1
L	1	0	1	1
M	0	0	0	0
N	1	0	0	0
O	0	1	0	0
P	1	1	0	0

X-10 key and function codes are represented by the sequential five-bit patterns shown in Table 1-2. You can distinguish key codes from function codes using the last bit, which is zero for key codes and one for function codes. The key codes correspond to what X-10 calls *unit codes* in most retail documentation.

Table 1-2 X-10 Key and Function Code Bit Patterns

Code	*Bit 0*	*Bit 1*	*Bit 2*	*Bit 3*	*Bit 4*
1	0	1	1	0	0
2	1	1	1	0	0
3	0	0	1	0	0
4	1	0	1	0	0

Continued

Table 1-2 *(continued)*

Code	*Bit 0*	*Bit 1*	*Bit 2*	*Bit 3*	*Bit 4*
5	0	0	0	1	0
6	1	0	0	1	0
7	0	1	0	1	0
8	1	1	0	1	0
9	0	1	1	1	0
10	1	1	1	1	0
11	0	0	1	1	0
12	1	0	1	1	0
13	0	0	0	0	0
14	1	0	0	0	0
15	0	1	0	0	0
16	1	1	0	0	0
All Units Off	0	0	0	0	1
All Lights On	0	0	0	1	1
On	0	0	1	0	1
Off	0	0	1	1	1
Dim	0	1	0	0	1
Bright	0	1	0	1	1
All Lights Off	0	1	1	0	1
Extended Code	0	1	1	1	1
Hail Request	1	0	0	0	1
Hail Acknowledge	1	0	0	1	1
Pre-Set Dim	1	0	1	X	1
Extended Data (Analog)	1	1	0	0	1
Status = on	1	1	0	1	1
Status = off	1	1	1	0	1
Status Request	1	1	1	1	1

If you've been planning how you'd use these codes as you read, you've noticed that you can't form a complete command in one transmission — that is, you can say things like `B12` or `B On`, but you can't write a message that says `B12 On`. What you have to do is send message sequences, such as `B12` followed by `B On`. The requirement to send each message twice with a three-cycle gap between messages results in the final message sequence:

```
B12 - B12 – B On – B On
```

This example shows that it takes at least four messages to complete a command, and of course transmission errors can corrupt any of the four. The specific behavior you'll see if one or more of the messages are corrupted depends on the devices involved, but all the effects are likely to appear as unreliable or erratic operation.

Because the addresses and commands are separate in the protocol, you can combine messages to give more complicated commands. For example, if you wanted to turn on both `B3` and `B12`, you could use the command sequence:

```
B3 – B3 – B12 - B12 – B On – B On
```

Many command codes have straightforward meanings; `On`, `Off`, and `All Lights Off`, for example, do just what you'd expect. `Status Request` commands the addressed unit to respond with `Status = on` or `Status = off` if it supports status responses.

Less straightforward are `Dim`, `Bright`, `Extended Code`, `Hail Request / Acknowledge`, `Pre-Set Dim`, and `Extended Data`. Here's what they do:

- **Dim, Bright, and Pre-Set Dim** — Standard X-10 incandescent lamp dimmers have 16 dimming steps. The standard protocol for dimming to a specific level is to turn the lamp on, then send a dim command pair (which would be two `Start code-house code-dim` sequences back-to-back with no intervening empty cycles) for each dim step. A sequence of 16 pairs should turn the lamp off completely. You can similarly send `Off` followed by a series of Bright commands to get the same effect.

  The requirement to start at full on or full off, then ramp the intensity to reach a known level produces awkward results. The Pre-Set Dim command was intended to address that problem by directing the dimmer to go to a specific level directly. A typical message sequence, one to set unit `A1` to a level, was

  ```
  A1 – A1 - <value>Pre-Set Dim - <value>Pre-Set Dim
  ```

  where `<value>` was a house code from `A` to `P`. If you look back at the encodings for the house codes, you'll see that used in the sequence M, N, O, C, D, A, B, E, F, G, H, K, L, I, J they correspond to the binary sequence 0, 1, 2, ... 14, 15. The sequence

  ```
  A1 – A1 - BPre-Set Dim - BPre-Set Dim
  ```

  would, therefore, set the dimmer to level six. X-10 apparently never made any products that implemented the Pre-Set Dim command, although some others did, so it's no longer defined in the protocol. (The sequence also violates the requirement that commands have the same house code as the prior address packets, potentially lowering the reliability of the system.) In the most recent documentation, X-10 has re-designated the Pre-Set Dim command code as being "for security messages." Products using Pre-Set

Dim will continue to work, of course, but as products come to market using that command code for security functions, your chances of conflicts go up should you use both categories of device.

- **Extended Code/Data** — You can do a lot more with the X-10 protocol than the basic function code commands in Table 1-2 permit, because the Extended Code and Extended Data function codes let devices send arbitrary information. Standard X-10 devices ignore everything in the message following those function codes up to the next gap (i.e., power line cycles with no tones in either zero crossing), so in theory extended messages can be any length.

  In practice, X-10 has defined specific extended message formats (see `ftp://ftp.X10.com/pub/manuals/xtc798.doc`). Extended message format 1 follows an Extended Code function, and consists of 4 bits of unit code, 8 bits of data, and 8 bits of command. Five types of interpretation are defined for the data and command bytes:

  - **Type 0:** Shutters and Sunshades
  - **Type 1:** Sensors
  - **Type 2:** Security
  - **Type 3:** Control Modules (Dimmers and Appliances)
  - **Type 4:** Extended Secure Addressing
  - **Type 5:** Extended Secure Addressing for Groups

  Nothing prevents you from using your own message formats with Extended Code or Data functions, except that repeaters may not properly echo your messages onto the other power line phases.

  The longer messages increase the probability that one X-10 transmission will collide with another, so the `xtc798.doc` specification adds some access requirements to the protocol. Specifically, it says Extended Code systems must follow specific steps to ensure the power line is not being used by another device, must check for collisions during transmission, and must respond to collisions by terminating the operation and starting over from the state in which it's verifying the power line is idle.

- **Hail Request/Acknowledge** — A device can send Hail Request to locate X-10 transmitters on the system. Those transmitters are to respond with Hail Acknowledge. The commands are used for other purposes by some equipment, and not implemented by all transmitters.

Implementation factors

The wiring in a house isn't a perfect transmission system. Worst of the transmission impairments the signal sees is the power transformer that feeds your house, as shown in Figure 1-6, because it splits the single-phase power into two sides, with a neutral tap between the two. Appliances see 120 V between a power rail and neutral, or 240 V, between the two power rails.

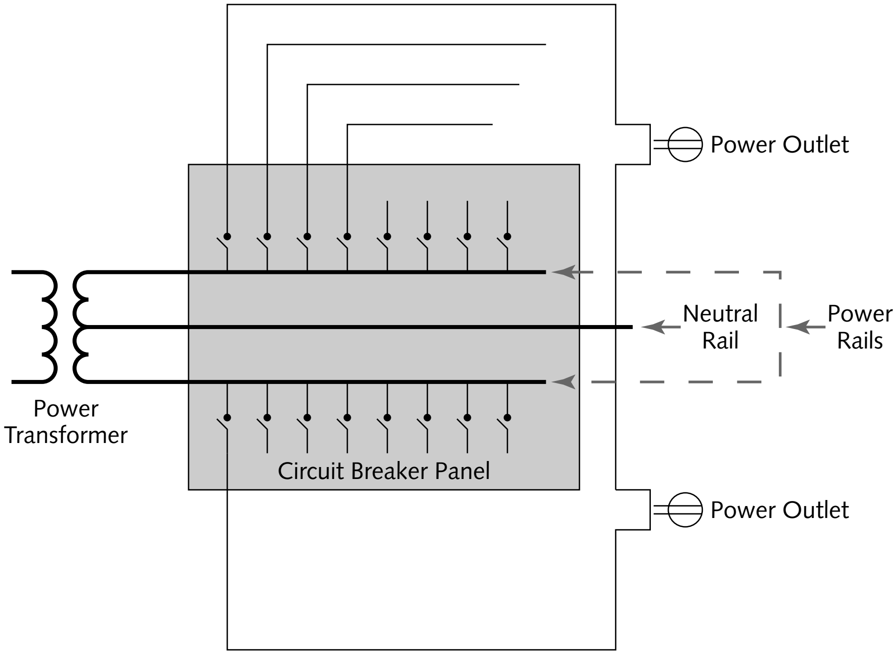

FIGURE 1-6: Home power system schematic

Suppose there's an X-10 transmitter connected to the upper power outlet in the figure and an X-10 receiver on the lower outlet. Signals output from the upper power outlet transmitter are referenced between the one power rail and the neutral, and are carried directly on those two wires to any receiver on the same circuit or with some loss to any circuit connected to the upper power rail. To get to the lower rail, however, the signal has to be coupled through the power transformer (or some 240-volt appliance) onto the lower rail. Power transformers are tuned to be efficient at 60 or 50 Hz, however, and their relatively high impedance at the 120 kHz signal frequency does not couple energy well — it's entirely possible that only a tenth of the original signal passes to the other side of the breaker panel. Even ignoring noise and other problems, should the coupled signal drop below the 100 mV (millivolt) reception threshold, receivers on the other side of the transformer won't work reliably, or won't work at all. You fix this problem using a *coupler*, a device that provides a low impedance path for the 120 kHz signal from one side of the breaker panel to the other to reduce the loss, or a *repeater*, a device that actually amplifies the signal before retransmitting it onto the other side.

Signal attenuation comes from other sources, too, because many types of devices on the power line will themselves attenuate the X-10 signal. Relatively small capacitances in power supplies and other electronic circuits look like high impedances to the normal 60 Hz power waveform, but may look like a low impedance to the 120 kHz signal and therefore reduce its strength considerably. The effects of each such small loss add together, too, so while no one computer, television, or other device may be enough to make your system unreliable, a large enough number of them may. You need a *filter* to fix this problem. Filters work as shown in Figure 1-7. Coils are the dual of capacitors, in that their impedance goes up with higher frequency, while a capacitor's impedance goes down. Putting a small inductance coil in series with each wire in the power line lets the power flow without hindrance, but blocks noise and the 120 kHz signal. The filter blocks any X-10 signal present on the power line on the left in Figure 1-7, keeping that signal from being attenuated by the low impedance capacitance in the device on the right.

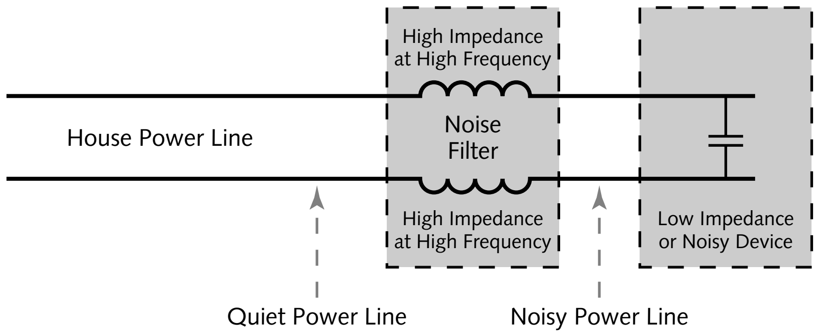

FIGURE 1-7: X-10 filtering

Noise also corrupts X-10 signals. The theoretical waveform of Figure 1-3 is what you think about, but what you get may be the mess in Figure 1-8 that results from even a moderate amount of noise on the power line. We've left the tones in, but no X-10 receiver is likely to figure out where they are and whether a zero or one is being sent. The figure is the same one as Figure 1-3, with random 10-volt peak-to-peak noise added on to the 120 volt power line signal.

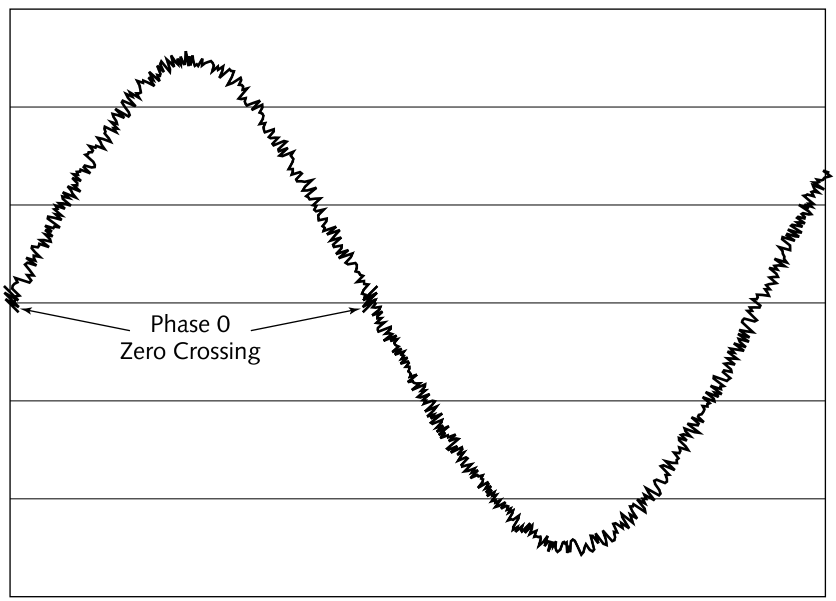

FIGURE 1-8: Power line noise

Devices that switch the power line on and off generate noise. Devices that generate noise that can affect your X-10 system include:

- **Power supplies** — Decades ago, power supplies used transformers to change voltage levels, then rectified and filtered the output to produce direct current. Such *linear power supplies* aren't very efficient, though, so the newer *switching power supply* design has almost completely replaced the older design. A switching power supply turns the power feed to a filter on and off very quickly, keeping just the right amount of charge in the input capacitor of the filter to maintain the precise output voltage. The switching operation generates noise back into the power line, however, noise that can affect X-10 systems if the power supply or an external filter doesn't block it. Everything electronic that doesn't run on batteries has a power supply.
- **Fluorescent lamps** — A fluorescent lamp contains an arc between two electrodes that dies and re-strikes at every voltage zero crossing of the power line. Each time the arc re-strikes, it generates wideband, high-frequency noise. Compact fluorescent lights are worse, because they contain switching power supplies, too.
- **Motors** — Many motors include a *commutator* and *brushes* to rapidly switch the incoming power among the different coils in the armature. Every time the commutator and brushes disconnect one coil and connect another, there's a surge of current that generates power line noise.
- **Your neighbor's X-10 system** — What's signal to your neighbor is noise to you, and if X-10 signals from adjacent houses couple in to your system, you'll see strange results as your equipment interprets and acts on commands leaking in from next door. The coupling is through the main power transformer and power lines feeding your house, so a filter on the mains is all you need to solve the problem should it occur.

Parts

There's a wide variety of X-10 components available as pre-built devices you can use to build systems. You can also build your own components using readily available building blocks. This section covers pre-built modules; the later section "Building Your Own Modules" shows how to use the building blocks.

Switches and dimmers

The simplest X-10 receiver is a plug-in appliance module, such as the Radio Shack module shown in Figure 1-9. It requires no special wiring concerns, and because it is either on or off, with no dimming capability, it can control any type of load up to its rated capacity.

The type of load you connect to a module is important for all other kinds of receivers. Resistive loads, such as heaters and lamps, do not respond to variations in the AC waveform. Reactive loads have capacitive and/or inductive components, in addition to the resistive load, that make the loads unsuitable for control by *triac* semiconductor devices that modify the AC waveform.

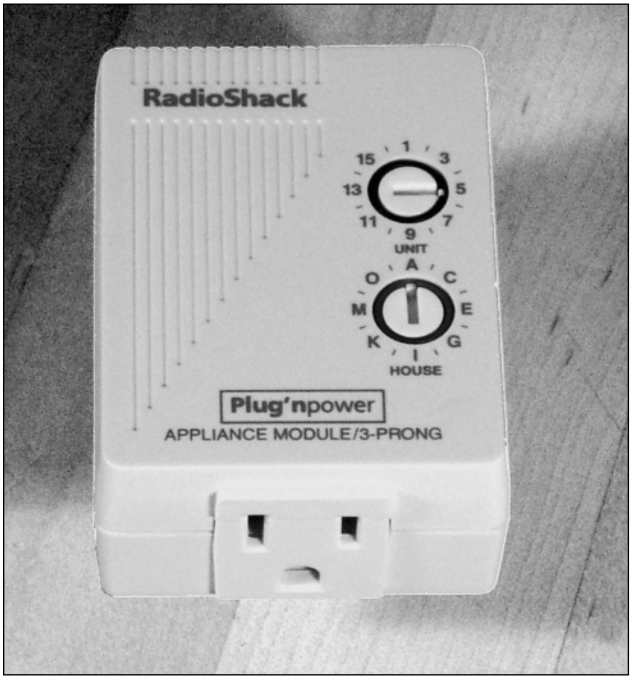

FIGURE 1-9: Appliance module

A triac is a three-terminal semiconductor device used by all modern dimmers to control an AC line. Two of the terminals form a path for the load current, while the third terminal turns the device on (the triac turns off at every zero crossing of the AC waveform). If the control terminal is always on, you get a standard AC sine wave. If there is a delay after the zero crossing before the control terminal turns on, you get a truncated AC waveform such as in Figure 1-10. Triacs replaced rheostats in dimmers because they're smaller and generate far less heat.

The frequency analysis in Figure 1-11 (derived from Figure 1-10) shows the problem the triac output waveform creates for reactive loads. The solid line is the fundamental frequency, while the dashed lines show the additional frequency components introduced by the waveform truncation. Reactive loads respond to those additional frequency components, and not necessarily in a good way.

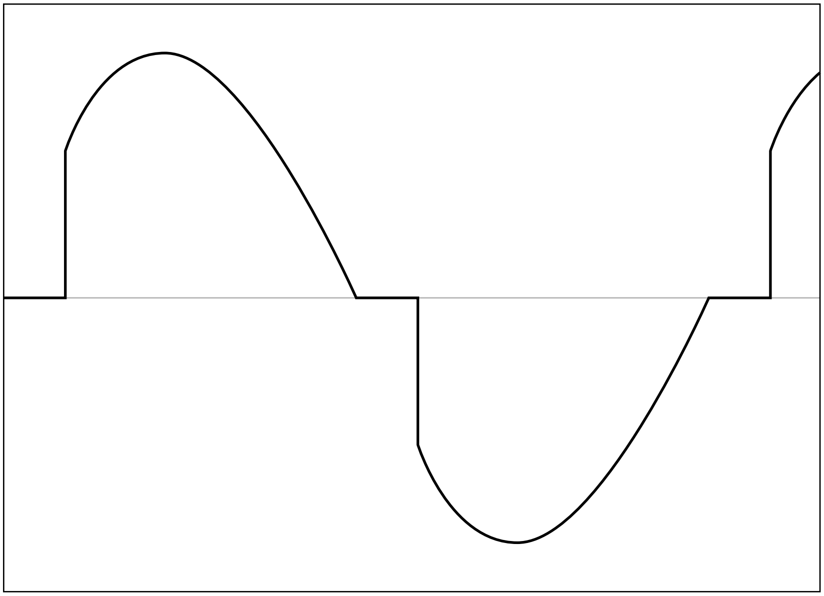

FIGURE 1-10: Dimmer waveform output

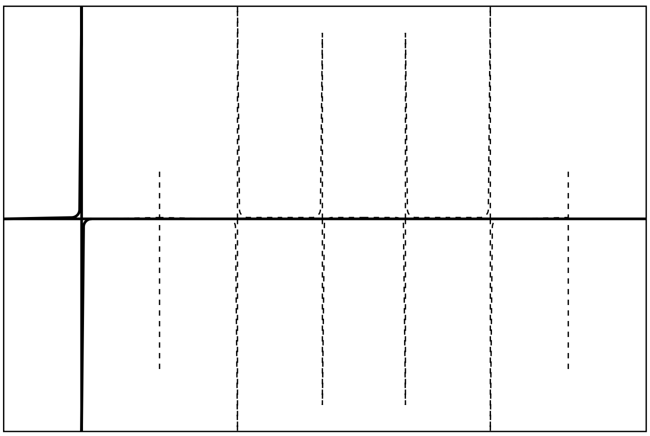

FIGURE 1-11: Dimmer output frequency analysis

The net result you need to remember from Figure 1-11 is that you must only use a dimming-capable X-10 module with a resistive load. Reactive loads—essentially anything but an incandescent lamp—require appliance modules rated for the more complex load. Even halogen lamps may have transformers in their circuitry, making them a reactive load and unsuitable for dimming.

Most X-10 light switches can dim, although the dimming function may only be accessible from a remote controller and not from the switch itself. Some, but definitely not all, can report their on/off status back to the controller.

Modules capable of replacing a single, standard light switch are not suitable for use in three-way or one-way configurations. Special module sets (Figure 1-12) are available for that purpose, and must be paired in a master/slave configuration.

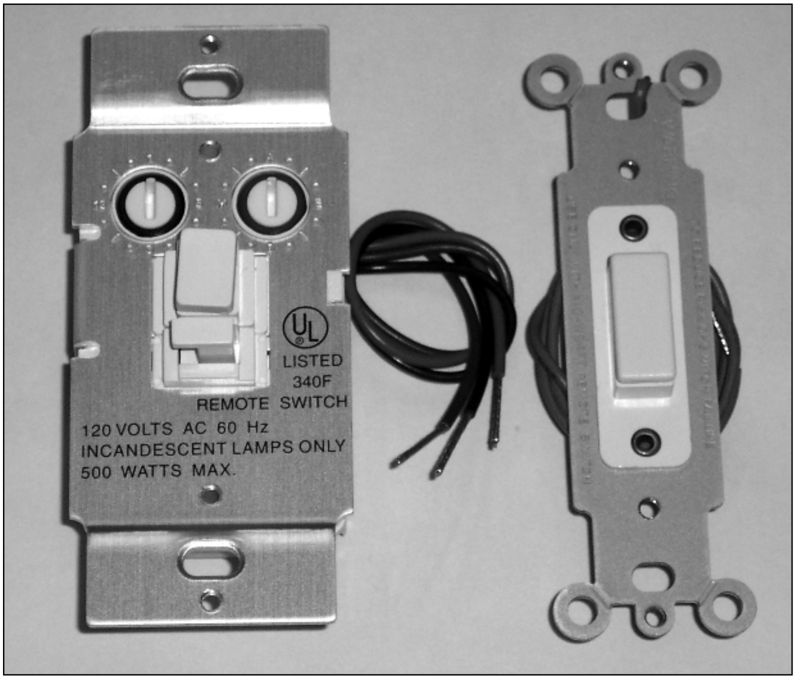

FIGURE 1-12: X-10 three-way light switch set

Controllers

There's a wide variety of controllers available to transmit X-10 commands over the power line, varying from simple table-top models to computer-controlled units. We've used the X-10 model CM11A PC controller, although many others exist. Table 1-3 lists a representative selection, all of which are controllable using the HomeSeer software. The Smarthome unit is particularly attractive because of its USB interface, helping you migrate away from serial ports.

Table 1-3 X-10 Compatible Computer Interface Controllers

Controller	*Source*
X-10 CM11A	X-10 `www.X10.com/automation/X10_ck11a.htm`
Smarthome PowerLinc USB	Smarthome `www.smarthome.com/1132U.html`
ACT TI103-RS232	Advanced Control Technologies, Inc. `www.act-solutions.com/pdfs/PCCSpecs/ti103_spec.pdf`

Although you'll typically use one of these controllers as the main transmitter in your system, controlling all your other devices through it, you don't have to restrict them to that role. You can have more than one in your system, and you can use them with a serial interface to devices like the Basic Stamp in the section "Building Your Own Modules" below or the Java Stamp we use in the Automated BBQ Temperature Control project (Chapter 6). That's a particularly attractive option with the Java Stamp, because (unlike the Basic Stamp) the Java Stamp does not have built-in routines to handle the low-level interface of the TW-523 Two-way Powerline Interface and related devices.

The protocol your PC uses to talk to a CM11A is defined in an X-10 document (`ftp://ftp.X10.com/pub/manuals/cm11a_protocol.txt`), including the information you'd need to use the interface with a Java Stamp.

The electrical connection to a CM11A is a standard RS-232 serial port, operating at 4.8 Kbps with 8 data bits, 1 stop bit, and no parity. The CM11A presents an RJ-11 connector; using the supplied serial cable, the signals are on the standard pins for a 9-pin connector (Signal In = pin 2, Signal Out = pin 3, Signal Ground = pin 5) plus one addition: the Ring In signal is used, and is on pin 9. The interface asserts Ring In when it needs service from the PC, such as when it hears a command or status message on the power line from another device. The interface can operate on any house or unit code.

Here's an example (Figure 1-13) of what you can do with just a CM11A and some switch modules. The HomeSeer software (`www.homeseer.com`; see Chapter 9) lets you enter your location, and from it calculates times for sunrise and sunset. Rules in HomeSeer then turn the lights in front of your house on at sunset and off at sunrise, with small random changes backwards or forwards each day to avoid making it look like the lights are on a timer.

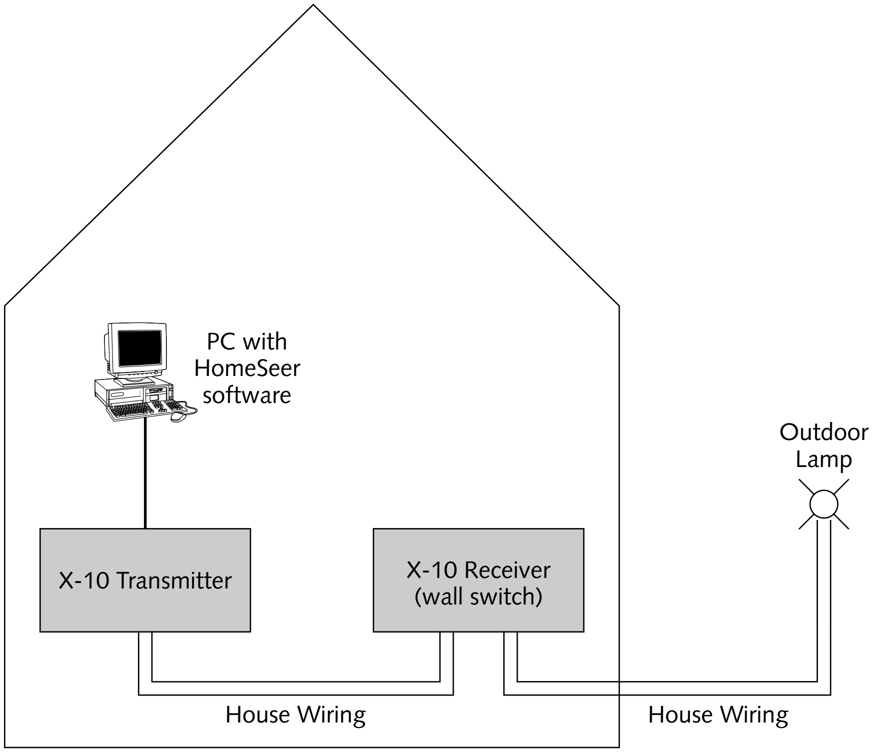

FIGURE 1-13: Outdoor lighting control

Wireless controllers and switches

Power line communications requires a power line connection, of course. It's inconvenient to walk around trailing an extension cord, which led to the creation of modules that let wireless devices interact with X-10 networks. The X-10 model TM751 Wireless Transceiver Module provides the interface between wireless units and the power line network, receiving wireless signals and generating the appropriate messages onto the power line.

One restriction you need to keep in mind using the TM751 is that, although you can select the specific one, the device only responds on a single house code. That's an advantage in applications where you have a large area to cover wirelessly. At most, you're only going to get approximately a 100-foot range between a wireless transmitter and the transceiver, and the range can be severely reduced by intervening metallic surfaces. You can usually solve the range problem by adding more transceivers, but you'll want them on different house codes to ensure they don't both receive a transmission and output to the power line.

Two alternatives to the TM751, shown in Table 1-4, give you some other options. The Smarthome All Housecode X-10 RF Receiver works at longer ranges, and can be programmed through your PC to operate on multiple house codes. The Smarthome X-10 RF Receiver to RS-232 Adapter gives you a direct wireless interface to your PC, giving you more options for wireless control of software such as HomeSeer.

Table 1-4 Wireless Transceivers

Transceiver	*Source*
X-10 TW751	X-10 `www.x10.com/automation/x10_tm751.htm`
Smarthome All Housecode X-10 RF Receiver (also requires X-10 TW523 or Smarthome PowerLinc)	Smarthome `www.smarthome.com/4831.html`
Smarthome X-10 RF Receiver to RS-232 Adapter	Smarthome `www.smarthome.com/4832.html`

An interesting problem you can solve wirelessly with X-10 control is putting a switch where there presently are no wires. The Smarthome X-10 Wireless Wall Switch enables you to do that—it looks like a four-element wall switch, but instead of wiring into a switch box in the wall, it transmits wireless codes to a nearby transceiver. You can let the transmitted codes control X-10 lamp or switch modules directly, or you can receive the transmission at your PC through a controller and execute any actions you desire.

Motion and light sensors

You don't always want to have to push buttons—wireless or not—to make something happen. Instead, as with outdoor security lights, you may want to trigger actions based on light (or its absence) or motion. Motion sensors (Table 1-5) solve that problem, sending messages for both motion and light. The X-10 MS14A is a wireless motion and light sensor you combine with a wireless transceiver, while the wired PR511 combines outdoor lamps with a motion and light sensor.

Table 1-5 Wireless Transceivers

Sensor	*Source*
X-10 Eagle Eye Motion Sensor	X-10 `www.x10.com/automation/x10_ms14a.htm`
X-10 Dual Motion Monitor (outdoor lamps with motion sensor)	X-10 `www.x10.com/automation/x10_pr511.htm`

Figure 1-14 shows a sample motion sensor application — controlling overhead lights in a garage. We placed three motion sensors, to make sure people were seen when they entered the space, and located the transceiver so there would be direct line of sight from the wireless motion sensors (we used X-10 MS14A sensors) to the transceiver. If you were to move the transceiver to an outlet directly above the metal shelves, for example, the path from the sensor on the bottom wall would be blocked and the sensor would operate only erratically, confusing the system when `Off` messages were dropped.

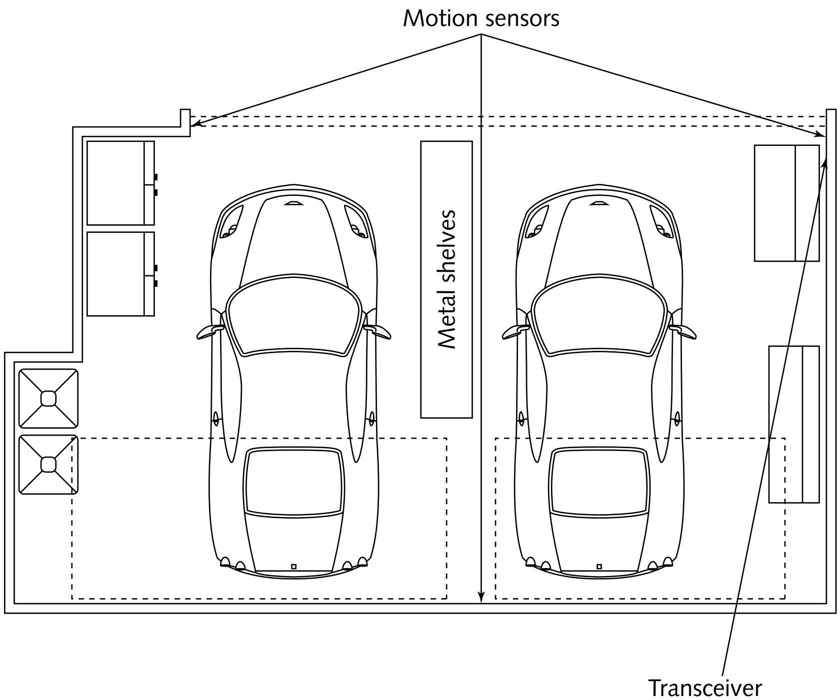

FIGURE 1-14: Motion sensors for automatic lamps

Low-voltage sensors and relays

Some X-10 applications require that you sense the status of a remote circuit, or control a device, but are strictly on/off in nature, and only involve relatively low voltages. Specific modules are available to meet those requirements, as listed in Table 1-6. The Universal Module lets you control devices needing a low power, low-voltage contact closure. You'll see an application of a Universal Module in the chapter on Automated Sprinkler Control (Chapter 5), where we've used one to enable or disable the 24-volt drive to the valves by a sprinkler controller.

Table 1-6 Low-Voltage Modules

Sensor	*Source*
X-10 Universal Module	X-10 www.x10.com/automation/x10_um506.htm
X-10 Powerflash (also Burglar Alarm Interface)	X-10 www.x10.com/security/x10_pf284.htm

The Powerflash does the inverse of the Universal Module, sending an X-10 message out on the power line when a contact closes or opens. You could use it to detect a switch closing and tell your PC when the garage door is open or closed; we used one in *PC Toys* (Wiley, 2003) as part of a system that warns you when a freezer gets too warm.

Filters, couplers, and repeaters

As Table 1-7 demonstrates, there's a wide variety of signal improvement and noise suppression equipment available. The devices in the table, many of which have both wired and plug-in versions, have these functions:

- **Noise Filter** — A simple noise filter works along the lines of Figure 1-7, keeping power line noise from crossing the filter in either direction. You can use one to isolate a noise source from the rest of your system, or to isolate a malfunctioning device from high levels of noise on your overall system. Some noise filters allow signals at the X-10 120 kHz frequency through, but block nearby frequencies, while full-spectrum filters block frequencies above the power line 60 Hz.
- **Noise Block** — A noise block is no different from a full-spectrum noise filter in principle, but it is packaged to make it convenient to install it right at the noisy equipment.
- **Signal Bridge** — Also known as a *passive phase coupler,* a signal bridge transfers signals from one side of the electrical panel to the other. A passive coupler divides the available signal energy, lowering the signal on the sending side and increasing it on the receiving side.
- **Blocking Coupler** — A blocking coupler, which is also a passive coupler, blocks entering X-10 signals as well as coupling signals between sides.
- **Phase Coupler / Repeater** — You can boost the signal strength of the receiving side of your system with a phase coupler/repeater. The device actually receives and re-transmits the received signal, outputting it with far more strength than was received.

Table 1-7 Filters, Couplers, and Repeaters

Sensor	*Source*
FilterLinc Plug-In Noise Filter	Smarthome `www.smarthome.com/1626.html`
Leviton Noise Block	Smarthome `www.smarthome.com/4835.html`
Leviton Signal Bridge	Smarthome `www.smarthome.com/4815.html`
Leviton and X-10 Whole-House Blocking Couplers	Smarthome `www.smarthome.com/4850.html`
SignaLinc Plug-in Coupler-Repeater	Smarthome `www.smarthome.com/4826.html`

X-10
Phase Coupler / Repeater
	X-10
	`www.x10pro.com/pro/catalog/`
	`xpcp.xpcr.page5.html`

X-10
XPF 3-Wire Noise Filter
	X-10
	`www.x10pro.com/pro/catalog/`
	`xpf.xppf.page5.html`

You have to be careful when installing filters to ensure that you don't isolate X-10 transmitters or receivers from the network by blocking their signals. Noise blocks, for example, must be installed between the controlling receiver and the load, not on the power line side of the receiver.

It's also important to avoid using multiple couplers or repeaters, because they're not designed to work as multiples, and can either interfere with each other or (worse) echo signals endlessly until you disconnect one of them.

Troubleshooting

We like to start troubleshooting problems by thinking about what might plausibly go wrong. X-10 networks are simple enough in operation that there are only a few options:

- **No signal reached the power line** — The transmitter might be defective, or the transmitter might be disconnected from the power line.
- **Unreliable wireless operation** — The X-10 motion detectors are the most common wireless transmitters; if their signal isn't received, the system will behave as if the sensor isn't working. Poor wireless signal reception can be caused by weak batteries, competing radio signals acting as noise and jamming the signal, blockage between transmitter and receiver, or excessive range between transmitter and receiver.

- **No signal, a weak signal, or a noisy signal reached the receiver**—There might be filters or other losses on the power line, the signal might not be coupling into the other side of the power panel adequately, or noise signals might be corrupting the incoming signal. Noise sources include contention between multiple X-10 transmitters operating at the same time.
- **The receiver isn't responding**—The receiver might be broken, might not be properly connected to the power line, might be receiving on an address different than the transmitter is sending, might not respond to an unusual command sequence being sent, or might be connected to a load that doesn't work.

Figure 1-15 shows that problems can afflict every part of the network, but because of the serial nature of the connections, shows that you can troubleshoot by starting at one end and following the signal (or its absence). You can start at either the controller or the load.

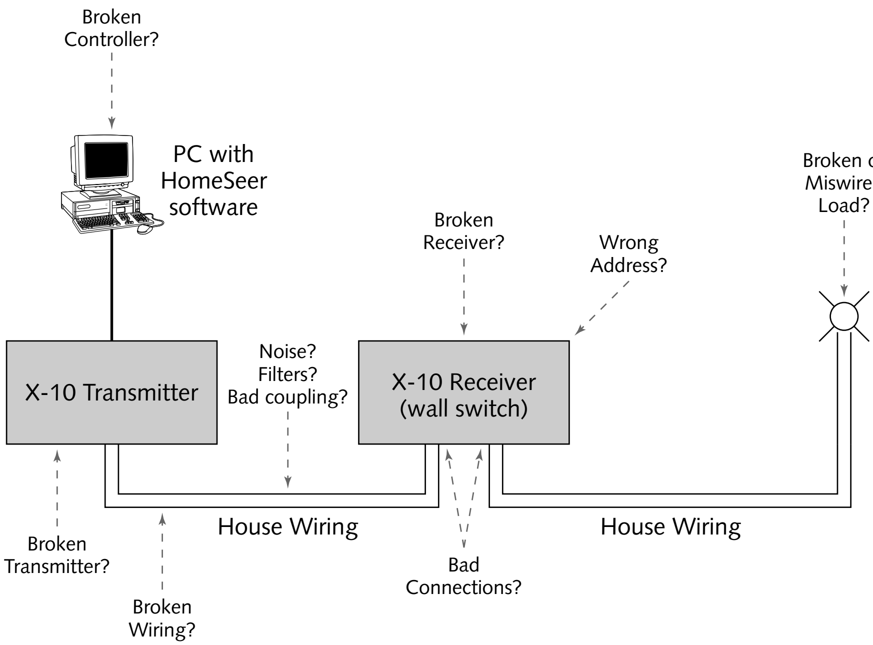

FIGURE 1-15: X-10 network faults

If you start at the load, your sequence might follow this sequence:

1. If there's a manual switch at the receiver, see whether it will turn the load on or off. If not, substitute another receiver temporarily and see whether it controls the load where the proper one will not.

2. Assuming the receiver can control the load, try isolating the receiver and a known good test transmitter on a power line isolated by a filter from the rest of the house, and see whether the transmitter will control the receiver. If not, check the addressing and connections, and perhaps replace the receiver.

3. The receiver working in isolation suggests that either the problem is external to the isolated segment or the receiver is not as sensitive as it needs to be. Use a signal strength meter, such as the combination of the X-10 Pro XPTT transmitter and XPTR receiver, respectively. The XPTT sends a continuous signal turning address `P1` on and off; the XPTR measures the strength of the received signal for `P1`. If you connect the XPTT at the location of your transmitter and the XPTR at the receiver, you can measure the signal transported by the power line between the two sites. You need 200 mV (millivolts) by specification, but if the signal is that low you'll need a very clean, noise-free system.

   If the problem *is* noise, you'll need to try to identify the source. Don't just turn units off, unplug them — many electronic designs don't really turn off even when they look off, and noise or loading from the power supply won't vanish unless you disconnect the power supply from the power line. Should you find a significant noise source, use a filter or noise blocker to clean up the pollution.

4. Use a known good receiver in an isolated segment against the transmitter, and try a test transmitter instead of the real transmitter.

5. Finally, although unlikely, erratic operation could be caused by signals fed in through the power transformer from a neighboring house. Use the XPTR to look for signals on the line when nothing should be happening.

It's not too likely you'd see unreliable operation from network collisions unless you're operating specialized equipment such as drapery or blind controls, but you can use the XPTR to see traffic loads. Figure 1-16 shows the X-10 Pro XPTT and XPTR units (`www.x10pro.com/pro/catalog/x10tools.html`). There are no controls on the XPTT; the single LED on the faceplate flashes when the device is in operation.

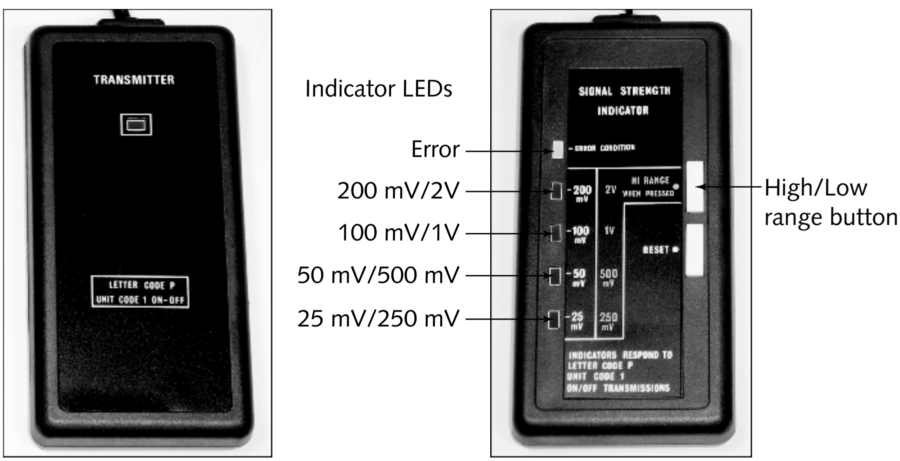

FIGURE 1-16: X-10 Pro XPTT (left) and XPTR (right)

The XPTR faceplate is organized into columns. The left column has LEDs to indicate a reception error or one of four signal levels; the right column has the Hi/Low Range button and the Reset button. Pushing Hi/Low Range sets the LEDs to indicate 250 mV to 2 V; releasing it selects the 25 to 200 mV range. There are no other controls — you just plug them into the outlets you want to test.

Home Automation with Standard Modules

The key requirement for wiring X-10 modules into your home is that they always have a connection between the hot and neutral power lines. Figure 1-17, for example, shows how you connect a basic dimming receiver to a light. The receiver connects in *series* with the light, not across the power line in *parallel*, and should be in the hot line, not the neutral. If you connect it in parallel, you'll blow the receiver the first time you turn it on.

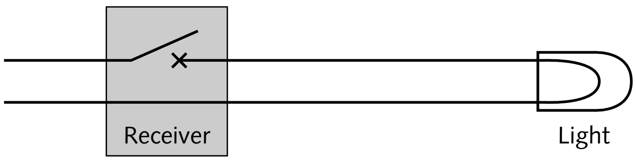

FIGURE 1-17: Basic receiver/lamp circuit

The receiver draws operating power from the circuit through the lamp. The light has a much lower resistance when it's off than when it's on, typically by a factor of 10 or more, so as the equivalent circuit of Figure 1-18 shows, there's less voltage drop across the receiver when on. The X-10 signal drops proportionally, so there's a smaller X-10 signal available to the receiver when the light is on. The effect is worse for small-wattage lamps — night-light bulbs have such a high on resistance that reliable X-10 control with a two-wire dimmer is unlikely. This effect causes `Off` commands to be less reliable when the X-10 signal strength is marginal, a condition you may be able to correct with an amplified repeater.

When you must control a small-wattage lamp, or if the system delivers marginal signal strength, the reduction of the X-10 signal to receivers in the on state can account for units you can turn on but not turn off reliably. You have to boost the received signal to solve that problem, or convert the installation to use a three-wire dimmer (Figure 1-19; don't confuse a three-wire dimmer with a three-way dimmer). Connecting the three-wire receiver directly across the hot and neutral wires lets it receive the X-10 signal unaffected by the on or off state of the lamp.

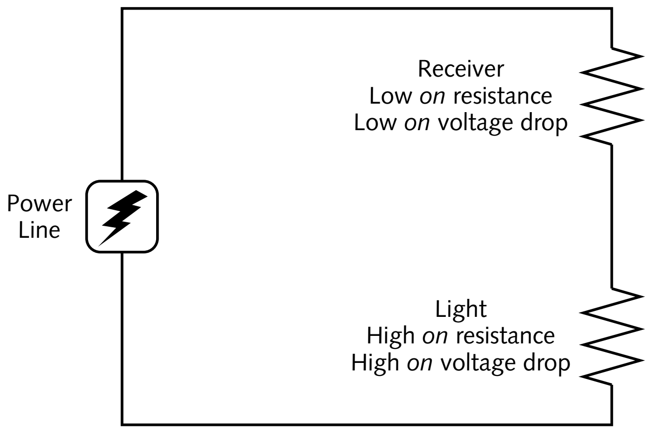

FIGURE 1-18: Equivalent circuit — light on

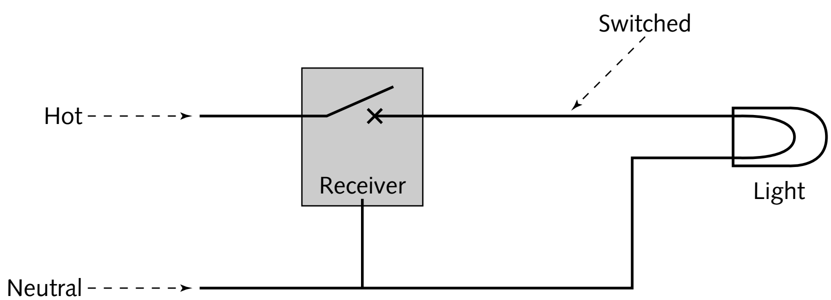

FIGURE 1-19: Three-wire dimmer/lamp circuit

Perhaps the most confusing X-10 installation topic is the requirements for wiring X-10 dimmers into a three-way circuit — that is, a circuit with two switches controlling a single light. Figure 1-20 shows the one way to wire conventional switches into a three-way circuit. The light will be on if both switches are up or are down, and off otherwise.

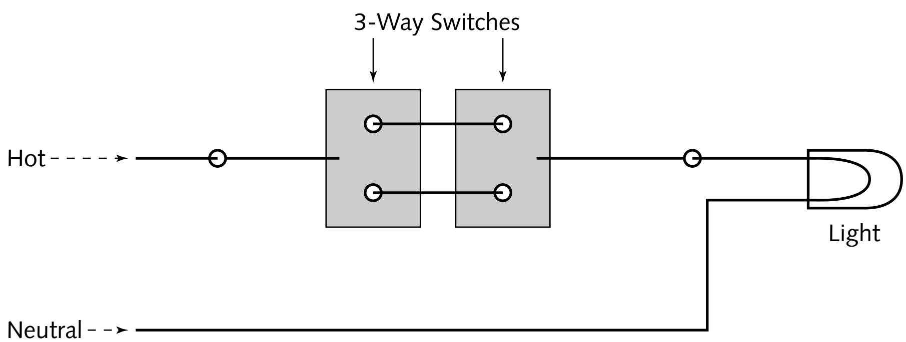

FIGURE 1-20: Simple three-way circuit

Unfortunately, the circuit in Figure 1-20 will not work if you simply replace one of the three-way switches with an X-10 dimmer, because for the wrong setting of the second three-way switch, there's no complete circuit through the light, and the dimmer gets no power. Because it gets no power, it never sees the signal to turn on. For that reason, X-10 three-way circuits are *entirely different* than the mechanical equivalent of Figure 1-20. An X-10 dimmer must connect directly to the light at all times, so the fundamental circuit has to be the same as the standard dimmer circuit of Figure 1-17. The second switch no longer controls the primary circuit; instead, it signals the dimmer. Figure 1-21 shows the idea — the second switch drives a control input to the dimmer, telling it to turn on or off by a pulse on the input. The light is always under control of the dimmer, and never directly under control of the slave.

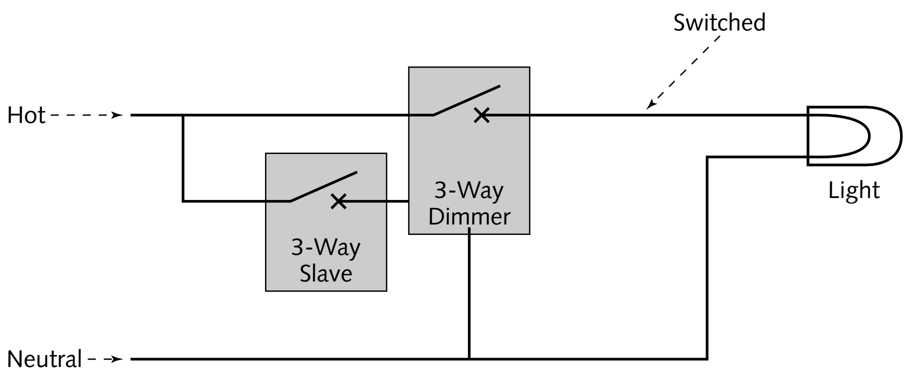

FIGURE 1-21: Basic three-way X-10 circuit

If you imagine the two wires between the switches in Figure 1-20 as corresponding to the two wires in the gap between the slave and the dimmer in Figure 1-21, you can see how you might wire the dimmer/slave pair in practice. There are a lot of different ways to wire three-way circuits, however; for more detail and explanation, and for how to extend the idea to one-way circuits, see the Advanced Control Technologies Web pages at `www.act-solutions.com/kingery06.htm` and `www.act-solutions.com/kingery07.htm`.

Of course, all of that is the hard way. If you simply install X-10 switches that output signals to the power line, you can then use your PC to signal any number of devices to turn on, off, or dim, regardless of the actual connections (or lack thereof) between the switches and anything else. No X-10 design problem has only one answer.

Building Your Own Modules

The X-10 TW523 is a two-way power line interface you can drive with your own circuits. It operates at a much more primitive level than PC interfaces such as the CM11A, which has a serial port interface, so you'll need both good electronics skills and experience with real-time embedded software. The interface gives you the following opto-isolated signals (Figure 1-22):

- **Zero Crossing** — Output from the TW523, the Zero Crossing signal goes high at the start of each zero crossing interval. The signal is not a conventional data valid signal; see the transmit and receive signal descriptions. Use a pull-up resistor on the signal.
- **Signal Ground** — Pin 2 is the common reference for the other three pins.
- **Receive from Power Line** — This signal is high if the received signal from the power line is a one, and low if it's a zero. The signal is not valid until 200 >s after Zero Crossing goes high, and is only required to stay valid for 1 ms after Zero Crossing goes high even if Zero Crossing remains high. Use a pull-up resistor on the signal.
- **Transmit to Power Line** — Output a high on this line within 200 µ of Zero Crossing going high, and hold the line high, to output a one on the power line — that is, to turn on the 120 kHz oscillator. Use the same timing with a low output to output a zero. The line should be low if the interface is not to transmit data, since that keeps the oscillator off.

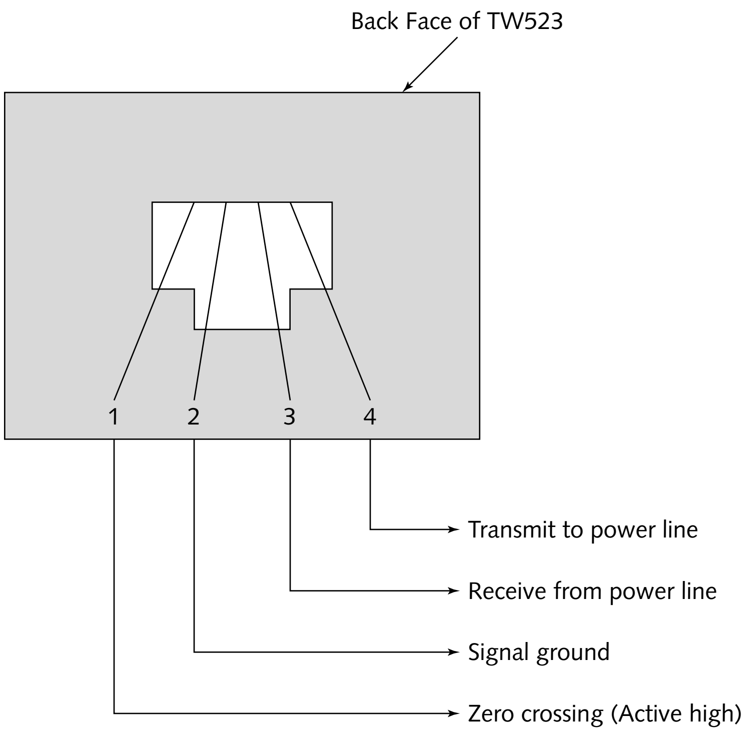

FIGURE 1-22: X-10 TW523 interface signals

Using the TW523 is somewhat difficult, because it provides only the physical layer interface, and requires compliance with relatively precise and demanding timing to ensure meeting the physical layer specification. The requirement to handle the physical layer timing plus all link layer requirements in your electronics implies you'll need some intelligence in devices you build using the TW523, not just sensors or switches. The Smarthome and Advanced Control Technologies interfaces in Table 1-8 relieve the requirement to handle critical timing in your implementation, substituting an RS-232 interface and more intelligence in the interface device, but you'll still need enough intelligence to handle the link layer. Those devices also add the capability to handle X-10 extended codes and data, whereas you can't receive the extended formats with the TW523.

Table 1-8 Two-Way X-10 Interfaces

Interface	*Source*
X-10 TW523	X-10 `www.x10.com/automation/x10_tw523.htm`
X-10 Pro PSC05	X-10 Pro `www.x10pro.com/pro/catalog/psc04.psc05.page12.html`
Smarthome PowerLinc II Serial/TW523 X-10 Interface with 12VDC Output	Smarthome `www.smarthome.com/1132.html`
Advanced Control Technologies TI103-RS232	Advanced Control Technologies `www.act-solutions.com/pdfs/PCCSpecs/ti103_spec.pdf`

A PC is far more elaborate than you want for a remote X-10 device; instead, you'll want to include an embedded microcontroller. We've used two different ones from Parallax in this book, as shown in Table 1-9. You can find many others available by searching the Internet; we chose the Parallax units because of their combination of capability, availability with boards supporting a circuit prototyping area, and integration with support software. The Basic Stamp runs a variant of the Basic programming language, while the Javelin Stamp runs an embedded subset of Sun Microsystems' Java.

Table 1-9 Embedded Microcontrollers

Embedded Microcontroller	*Source*
Parallax Basic Stamp Starter Kit with Board of Education	Parallax `www.parallax.com/detail.asp?product_id=27203`
Parallax Javelin Stamp Starter Kit	Parallax `www.parallax.com/detail.asp?product_id=27237`

Basic Stamp overview

The Basic Stamp (shown wired for the X-10 TW523 in Figure 1-23) is a small microcontroller programmed with firmware to interpret the Basic programming language. The Basic Stamp 2 (BS2) module we used includes 32 bytes of RAM, of which 6 are used for I/O and 26 for variables, and 2KB of EEPROM, enough to hold around 500 instructions. There's a comparison of all the Basic Stamp versions at `www.parallax.com/html_pages/tech/faqs/stamp_specs.asp`; other versions support more I/O pins, provide a scratch pad RAM, add more built-in commands, and hold much larger programs. They all consume more power, too.

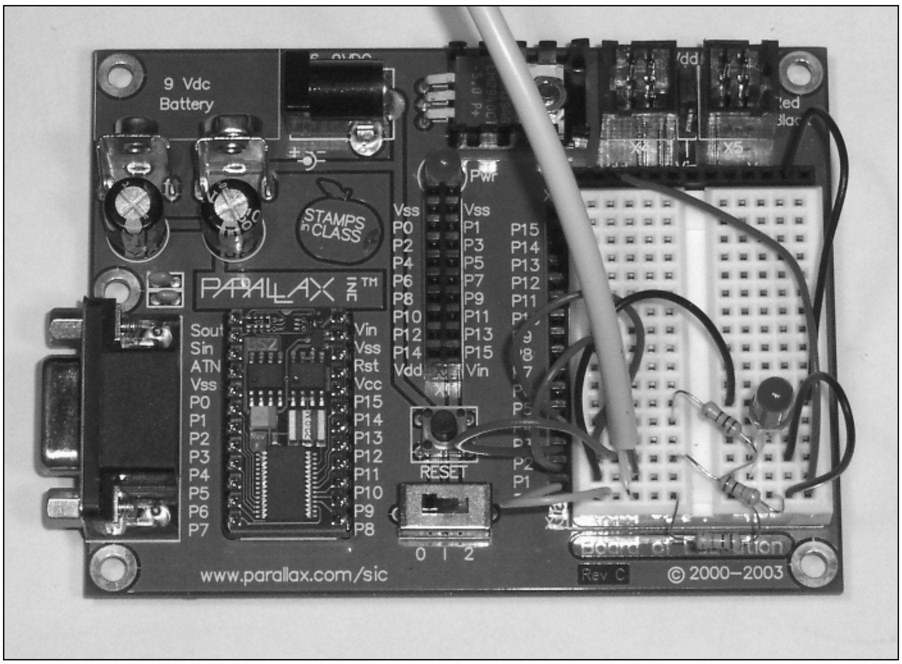

FIGURE 1-23: Parallax Basic Stamp

Performance of the Basic Stamps ranges from about 4K to 12K instructions per second, a range of about 3:1. We used the following program to make the performance measurements in Table 1-10, commenting out different parts of the program and changing variable types for the different tests. The version in the listing is the one used for the GOSUB benchmark.

```
'{$STAMP BS2}
'{$PBASIC 2.5}

Foo   VAR Byte
ix_1  VAR Word

DEBUG "Start", CR
FOR ix_1 = 0 TO 30000
  'Foo = 1
  GOSUB Sub
NEXT
DEBUG "Done", CR
END

Sub:
  RETURN
```

Table 1-10 Basic Stamp Performance (Stopwatch Timing)

Statement	*Time for 30K Iterations (s)*	*Time per Iteration (>s)*
Assignment (bit)	8	267
Assignment (nibble)	7	233
Assignment (word, byte)	6	200
FOR – NEXT (word variable)	22	730
GOSUB – RETURN	23	767

The Basic Stamp doesn't have a built-in real-time clock, so we timed execution with a stopwatch, adjusting the loop count to require 20–30 seconds total except for the GOSUB, which required about twice as long. Times shown in the table for the assignments and GOSUB tests are after subtracting off the 22 seconds required for the FOR-NEXT loop using a word variable. Depending on the actual program you run, the table results are reasonably consistent with the performance Parallax states for the BS2, which is about 4K instructions per second. In comparison, the Javelin Stamp Java processor runs far faster, at about 1 μ per FOR loop iteration — over 700 times faster than the Basic Stamp.

The BS2 performance is acceptable for relatively slowly changing work, such as user interfaces, weather readings, or process control calculations, but won't handle the faster demands of interfaces to A/D or D/A converters, or to many I/O devices. The TW523 is a good example — zero crossings occur a little more than 8 ms apart, leaving time for very few instructions, and fewer yet if you tried to organize your program with a modular structure involving loops and subroutines. The need for an interpreter running on the small embedded microcontroller to execute the Basic Stamp language accounts for the relative slowness when compared to the processor's 20 MHz clock. The Basic Stamp language compensates for the overhead of the interpreter by including several built-in commands to execute common embedded functions at much higher speed. Those built-in commands include ones to do A/D and D/A conversions, interface to serial communication ports and synchronous serial devices, generate DTMF tones used on telephone lines, and support raw interfaces to LCDs.

Build an X-10 test transmitter

One of the built-in commands in the Basic Stamp language, XOUT, supports X-10 physical layer output to a TW523 interface. The command synchronizes with the Zero Crossing signal input on one pin, and drives the Transmit to Power Line signal on a second pin. The following program drives a sequence of `P1 On` and `P1 Off` commands onto the power line, somewhat similar to the constant stream of `P1 On` commands output by the X-10 Pro XPTT. (Indeed, if you modified the program to have no PAUSE delays, the output would be essentially equivalent.)

```
'{$STAMP BS2}
'{$PBASIC 2.5}

Zpin     CON 0   'Zero Crossing on Pin P0
Mpin     CON 1   'Modulation output on Pin P1
HouseA   CON 0   'House code base
Unit1    CON 0   'Unit code base

TheHouse  CON HouseA+15       '+15 is P, for XPTR (wants P1)
TheUnit   CON Unit1           'e.g., +4 is Unit5

reps  VAR Word

  DEBUG  DEC ? TheHouse, DEC ? TheUnit

  FOR reps = 1 TO 100
    DEBUG DEC ? reps

    DEBUG "On", CR
    XOUT  Mpin, Zpin, [TheHouse\TheUnit]
    XOUT  Mpin, Zpin, [TheHouse\UNITON]                  'Turn it on
    PAUSE 1000
    DEBUG "Off", CR
    XOUT  Mpin, Zpin, [TheHouse\TheUnit]
    XOUT  Mpin, Zpin, [TheHouse\UNITOFF]                'Turn it off
    PAUSE 1000
  NEXT

  DEBUG CR, "Done"

  END
```

We investigated writing software to read and report X-10 commands with the Basic Stamp, but ultimately the device is too slow. The SHIFTIN command looked promising for a time, offering the capability to clock in data from a serial line, but because the Basic Stamp can only generate the clock pulse for SHIFTIN and not accept a clock from the external device, it's not usable with the TW523. So you'll understand our approach, the following program was intended to test X-10 input using direct pin I/O, but it does not work because it can't cycle fast enough to read the pins at the 120 Hz half-cycle rate. If you rewrote it in Java, the Javelin Stamp is likely to be fast enough.

```
'{$STAMP BS2}
'{$PBASIC 2.5}

' X-10 input example
' Geek House
' Software copyright (C) 2004 by Barry and Marcia Press

' ================================================================
' Data Declarations
' ================================================================
Dpin  PIN 2          ' X10 data input = P2
Opin  PIN 3          ' LED output = P3
X10Out  PIN 1        ' X10 command pulse output = P1
Cpin  PIN 0          ' X10 zero crossing input pin = P0

DataIn  VAR Bit     ' Sampled X-10 input data
ClockIn VAR Bit     ' Clock state when DataIn sampled
Error   VAR Bit     ' Non-zero if error found
Xbit    VAR Bit     ' X-10 sampled bit (not just data)
Xerror  VAR Bit     ' Aggregate address / command error

IXBits  VAR Byte    ' Loop counter for accumulating X-10 bits
Nbits   VAR Byte    ' Input parameter, GetBits bit cound
Xbits   VAR Word    ' Output result, GetBits
Xaddress  VAR Word  ' X10 address result
Xcommand  VAR Word  ' X10 command result
Ncmds   VAR Word    ' How many commands received

' ================================================================
' Main Program - X-10 input example
' ================================================================

  ' Ensure X-10 output pin inactive
  OUTPUT X10Out
  LOW X10Out

  ' Initialize output pins
  OUTPUT Opin
  HIGH Opin

  ' Initialize input pins
  INPUT Cpin
  INPUT Dpin
```

```
  DEBUG "Ready", CR

  ' Input and print 9-bit X-10 commands
  Ncmds = 0
  DO
    ' Read in the address and command packets
    Nbits = 9
    GOSUB WaitNoZC
    GOSUB GetBits         ' Address
    Xaddress = Xbits
    Xerror = Error

    GOSUB GetBits         ' Command
    Xcommand = Xbits
    Xerror = Xerror | Error

    DEBUG DEC Ncmds, " = "
    TOGGLE Opin
    DEBUG BIN Xaddress, " | ", BIN Xcommand

    IF (Error) THEN
      DEBUG " Error"
    ENDIF
    DEBUG CR
    Ncmds = Ncmds + 1

  LOOP

END

' ==================================================================
' Make sure not in zero crossing
' ==================================================================
WaitNoZC:
  DO WHILE (Cpin = 1)
  LOOP
  RETURN

' ==================================================================
' Read a sample from the next synchronous X-10 input zero crossing
' Returns data value in DataIn, clock for sample in ClockIn
' Sets Error non-zero if clock problem
' ==================================================================
GetSample:
  ' Wait for zero crossing interval to start
  DO WHILE (Cpin = 0)
  LOOP

  ' Get data line and sample clock for verification
  DataIn = Dpin
  ClockIn = Cpin
  Error = ClockIn ^ 1
  IF Error THEN
```

```
    DEBUG "."
  ENDIF

  ' Spin loop wait until not in zero crossing
  'DO WHILE Cpin = 1
  'LOOP

  RETURN

' ================================================================
' Wait until start code (1110) found on X-10 data line
' ================================================================
WaitStart:
  ' Wait for sequence of three ones. First one just hangs waiting for
  ' a one. Subsequent ones do not wait, but decode.
SequenceStart:
  DO
    GOSUB GetSample                     ' First sample
  LOOP WHILE (Error = 1 OR DataIn = 0)

  GOSUB GetSample                       ' Second sample
  IF (Error = 1 OR DataIn = 0) THEN SequenceStart

SequenceOnes:
  GOSUB GetSample                       ' Third sample
  IF (Error = 1 OR DataIn = 0) THEN SequenceStart

  GOSUB GetSample                       ' Fourth sample
  IF (Error = 1) THEN SequenceStart
  IF (DataIn = 1) THEN SequenceOnes

  'Found it, exit the subroutine
  RETURN

' ================================================================
' Get one bit from X-10 input.
'   Xbit is returned bit
'   Error is non-zero if receive error or the bit fails parity test
' ================================================================
GetBit:
  ' Get actual bit, exit if error receiving it
  GOSUB GetSample
  Xbit = DataIn
  IF (Error <> 0) THEN
    RETURN
  ENDIF

  ' Get complemented bit, error if receive error or not complement
  GOSUB GetSample
  Error = Error | (DataIn ^ Xbit ^ 1)
  RETURN
```

```
` ================================================================
` Get bits from X-10 input.
`   Nbits is number of bits to read
`   Xbits is returned bit sequence shifted in from low end
` ================================================================
GetBits:
  Xbits = 0
  GOSUB WaitStart
  FOR IXBits = 0 TO Nbits
    GOSUB GetBit
    IF (Error) THEN
      RETURN
    ENDIF
    Xbits = (Xbits << 1) | Xbit
  NEXT
  RETURN
```

Summary

The ability to send and receive signals across the power line, albeit at low rate, and to add both PCs and embedded microprocessors to the control system, gives you complete control over any powered device in your home. This chapter showed you the basic components you have to work with, and how to build systems using them. What you do with those elements is up to you and your imagination.

In Chapter 9 we'll show you how X-10 controls let you keep your yard watered just the way it should be, regardless of extraordinary heat or precipitation.