EPC Feature Article

Dennis Wilkison

# Home Scoping with X-10

Today, it seems like everybody is racing to put everything possible on the Internet in the form of web appliances. As Dennis explains, embedded programmers are learning to face the challenges and opening new avenues along the way.

Do you want to scope your house from the outside world? Home scoping is a simplified way to connect a web browser and its complex TCP/IP protocol to your X-10 modules. The X-10 protocol has been around for years, and interest in it is growing exponentially.

The 1970s brought the microprocessor revolution, the '80s unveiled the PC, and the '90s introduced a web of computers. This decade will witness web appliance invasion. Converting to web programming is a challenge faced by embedded programmers. At first, connections among computers and peripherals used proprietary hardwired interfaces. Then, modems and RS-232 allowed phone lines to be used for remote connections; increased modem speed and cost reductions made long distant communications possible. Enhanced protocols, high-speed backbones, and a need for broad information exchange created a web highway for traffic.

Some embedded programmers adopted web technology early in the game and demonstrated the 'Net as a connectivity medium for virtual scoping. Virtual scoping is a new term for using browsers to investigate web appliances. It means using a remote browser to visually interrogate a home page that is interactive with hardware.

I have used X-10 for years to control my home. The lights, wake-up alarm, and security controls haven't always made me popular. Forgetting to change the settings on the controller interface after a vacation can be embarrassing, especially sitting in a dark room after the controller turns out the lights. Now, I control those functions from anywhere. From the office, I can make the house ready for my return by turning on the heater, lights, and a pot of tea.

There is a gap between many embedded programmers and web browsers. Engineers often use a browser to locate information or buy and sell goods. But, many haven't used it to provide an interface with their embedded designs. In this article, I'll present an overview of the process. The included sidebar covers the basics of what makes embedded web servers tick. Then, you can develop web pages that can virtually scope the world.

The task seemed daunting at first. All those languages, TCP/IP, LAN networking, hardware, Ethernet, web connections, URLs, scripts, and so on were required before I could begin. But, the solution was simple when I discovered Advanced Web Communications' (AWC) Slim-Link Server module, support software, and evaluation board. Xecom's new division developed a way to circumvent the trials and pitfalls. Developing a dynamic home page to an X-10 interface with a house was fun.

| | |
|---|---|
| Bit 3–Bit 0 | Unit code |
| Bit 7–Bit 4 | House code (where house code A = 000B, B = 001B, and C = 010B) |
| Bit 12–Bit 8 | Future |
| Bit 15–Bit 13 | Action code (where on = 010B, off = 011B based on unit number, dim = 100B, bright = 101B, general commands without unit number requirements are: all lights off 000B, and all lights on = 001B) |

**Table 1**—*This table shows the X-10 information that is passed to the* `xten.cgi` *driver.*

**Listing 1**—*Check out the example of a button call to X-10.*

```
<html>
<body bgcolor="#F4D7C4">
<h1 align="center"><font color="#C92737" size="6" face="Impact">
  <em>Example of CGI calls</em></font></h1>
<div align="center"><center>

<table border="0">
    <tr>
        <td valign="top"><form action="Test.cgi" method="GET"
        name="memform" onsubmit="return CheckEntry()">
            <table border="0">
                <tr>
                    <td> </td>
                </tr>
            </table>
        </form>
         </td>
    </tr>
</table>
</center></div>
<p align="center"><input type="submit" value="Submit"></p>
<p align="center"><font color="#FF0000" size="1" face="Arial,
    Helvetica"> <b>©1999 - Xecom, Inc. - All rights Reserved</b>
</font></p>
</body>
</html>
```

## HARDWARE

I chose the AWC module concept because it provides an Ethernet connection via TCP/IP to the gateway. The module requires at least 600 mA and can be run off a battery and a solar cell. It measures 2.75 × 1.38 × 0.42 and fits anywhere. The connectors and interface to the outside are bigger than the module. The evaluation board was a natural choice due to the breadboard area (see Figure 1).

Unlike other servers, embedded web servers lack a moving platter hard drive. This usually means they have flash memory for the web pages and the operating system. Embedded web servers require compressed files and minimized code space, keeping graphics to a minimum. The missing mechanical quality in a hard drive allows embedded servers to go where desktop servers fear to tread.

The hardware functions of the Slim-Link (see Figure 2) consist of a microcontroller with communication and I/O ports. For this article, I'll use AWC86 as the example. The communication section consists of two RS-232 serial ports, an Ethernet port, and an RS-485 port. COM A and B are the RS-232 serial ports. The COM ports use autobaud for line speed detection.

A modem is an option on the AWC86. The AWC86A's modem connects to serial port COM B and connects to the outside world using a phone line. COM A connects to a terminal. The terminal needs to be set with disabled hardware controls because there are no RS-232 hardware lines (DSR, DTR, DCD) for this port. The connection to the Internet is made through the Ethernet LAN port J1. An RS-485 link is available for remote device serial connections.

The microcontroller is an AMD186 processor running with a 40-MHz clock. The firmware powering this web site is stored in the 512-KB flash memory at the factory. You can reprogram the flash memory to include application code. The program is downloaded from the flash into the SRAM at powerup, and runs from the SRAM. The hardware interface part of the system uses 32 ports to communicate with, control, and measure the world attached to the server.

An optional XE5614 modem that replaces COM B is provided with AWC89A. This uses six of the 32 ports, two for data and four for modem hardware handshaking. The complete 56,600-bps modem supports data compression, error correction, and fax transfer. This unique feature allows transmission of fax messages from the web server to any location in the world. It also provides for connections to pagers and other dial-up services.

The dial-in capability allows for a backdoor connection if the web is down, or it can be programmed for a

**Listing 2**— *Here's a CGI subroutine tool kit example of a password.*

```
/*************************************************************
*CGI Subroutine: Asking for the password using (guest information)*
*************************************************************/
    void form_ask_guest_pw(void)
    {
    web_tit("X10 Module demonstration","Entrance to Floor
Plan",get_bgcolor(),NULL,NULL);
    XeWebSend("<table><tr><td nowrap>"
     "<form method=\"GET\" action=\"guest.cgi\" name=\"memform\">"
     "Input Password:<br><br>"
     "   <input type=password name=pw size=9>\n"
     "<input type=submit value=Submit>  </form>"
     "<br><br><br><br><br><br><br><br>"
     "</td><td valign=top>"
     "<table border=1 width=100% bgcolor=");
XeWebSend(get_bgcolor());
XeWebSend("><tr><td><blockquote><font face=Arial size=2"
     " color=#000080><br>"
     "The Floor plan is pre-set in the Compile code. "
     "The House Code is set by using the Console. "
     "<ol>"
     "<br><br>"
     "</font></blockquote></td></tr></table></td>"
     "</tr></table>\n");
      web_idx();
    }
```

security measure to change the embedded server code. AWC is developing a PPP protocol connection for the module that will eliminate the need for a dedicated Ethernet connection to the Internet. A browser can dial in to this PPP port directly, connecting to the stored web pages.

You control the remaining 26 user-programmable input, output, time, and serial ports. They consist of two timers (four pins), one serial port with no hardware controls (COM A, two pins), two multifunction PIO/IRQ ports (two pins), eight PIO ports (eight pins), eight A/D converter inputs (eight pins), and two D/A outputs (two pins).

The D/A and A/D bit resolution is 12 bits, giving enough range for most applications. Several of the 22 digital I/O lines can be used to connect to a more complex converter. This would allow specific applications to use more resolution or dynamic range than that offered by the 12 bits in the internal converters. A temperature probe or thermistor could be added and monitored from this port.

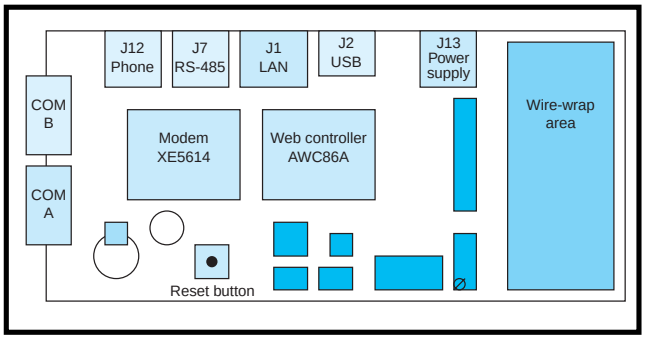


**Figure 1**—*Here's a closer look at what comes with the AWC86A evaluation board.*

## TW523

I needed an inexpensive hardware method to connect the AWC module to the X-10 protocol. An excellent article, "A Temperature-Sensing Control Device" by Donald Blake (*Circuit Cellar* 113), described the tone sequence X-10 uses around the zero crossing of the power line. A serial port driver version connected to the CP290 or the CM11 hardware also could have been used. The TW523 costs $15 to $20 and the CM11 is closer to $100. The interface driver for each has to be written or at least modified for the AM186 processor.

The code in Ken Davidson's article (*Circuit Cellar* 3) provided a good starting place. It has detailed information about the IBM PC version of a driver that connects to a parallel port. It wiggles the port lines according to the X-10 specification. Figure 3 shows how each of the two X-10 interfaces can be connected to the Slim-Link.

There's a specification for the TW523 to help you understand how the X-10 protocol communicates throughout the house wiring (read the TW523 manual at smarthome.com). Reading past *Circuit Cellar* articles, there is more than enough information to rewrite the driver for the Slim-Link. The code for the TW523 driver is now part of the AWC library. AWC is collecting a significant number of useful drivers, simplifying the interface to standard modules.

## HARDWARE INTERFACE

TW523 uses several components to interface with a TTL-level microprocessor. These components were mounted on the Slim-Link Server evaluation breadboard area. The interface consists of three circuits that provide level shifting and diode protection for the AWC module. Figure 4 shows the components and values needed according to the X-10 Powerhouse specification. You may find the TW523 interface manual at www.X-10.com/support/support_manuals.htm.

The AWC Slim-Link Server evaluation module provides the necessary hardware to interface between a LAN on the Internet and the X-10 specification required by the TW523 module. The last hardware necessary is a cable (standard RJ11 telephone extension cable) that connects the interface circuit to the TW523. Figure 4 also shows the necessary connection points on the AWC Slim-Link Server module.

The prewritten driver required the use of pins D20, D21, and IRQ1 on the AWC module for the TW523 interface. The zero crossing needed a high-level interrupt to ensure synchronization with the power line; IRQ1 was the best choice in the real-time operating system (RTOS).

A house code must be selected and set with each transmission for the interface to X-10. I discovered the AWC module has provision for special variables that are stored in flash

**Listing 3**— *Here you see the CGO subroutine for a basic password page.*

```
    /*************************************************************
    *

    *    CGI Subroutine: Create the page verifying password
    *
    *************************************************************/
    void form_askpw1(void)
    {
      web_tit("Visitor","Entrance to Floor plan","House Code
A",NULL);
      XeWebSend("<table><tr><td valign=\"top\" nowrap>"
        "Input Password:<br><form method=\"GET\" "
        "action=\"house_pg.cgi\" "
        "name=\"memform\">"
        "  <input type=password name=pw size=9>"
        "<input type=submit value=Submit>  </form>"
        "</td><td><table border=1 width=100%
bgcolor=#FAF5BE><tr><td>"
        "<font face=Arial size=2 color=#000080><blockquote><br>"
        "A password is needed to access the site. "

       web_idx();
    }
```

### The Basics

Let's start at the beginning. The world wide web is just a network—an immense network. In a local area network (LAN) a hub is cabled to a file server and all work stations. The network can expand by connecting hubs together with routers, developing a wide area network (WAN). The next level connects WANs with high-speed backbones. This hardware needed a language to communicate and unique addresses for each workstation, hub, and WAN, hence the Internet.

Local access points, called point of presence (POP), provide dial-in access. The user workstation communicates using the Internet Protocol (IP). Each connected computer runs a copy of IP to communicate with other connected computers or servers. Each IP has a unique address composed of four sets of numbers (maximum of three digits) separated by periods. A unique number is assigned to each telephone instrument.

The world wide web is the most popular use of the IP. It uses web browsers to move information from the selected server to the client. I'll use this part of the Internet to provide a virtual connection to scope the X-10 system in the house.

The Internet relay chat allows conferencing over the network, and the NFS (network file system) interconnects files among hosts. Telenet allows for a remote login to a host. As the world wide web changes, there will be more paths to hardware under virtual control.

The AWC module provides a clear way to understand CGI (Common Gateway Interface) and HTML and how they interact in an appliance web device. The first issue is how everything connects (see Figure 8).

Many embedded programmers have little experience in web programming, especially active or dynamic web pages. HTML is not complicated for writing static web pages. Author tools help you develop a web page based on presentation so you don't have to program in C or CGI. The document displayed on the browser is stored in the server flash memory as HTML.

Programs like Microsoft FrontPage are expensive, but less expensive programs like WebPage Construction Kit 5.0 work well, too. Later versions of Netscape allow access to HTML code and provide for modification. And, several HTML editors are on the Internet as shareware.

### Static and Dynamic Web Pages

There's an important difference between a static and dynamic web page. Changing pages that have information you can't type ahead to the server web site is a problem unless you use a dynamic web page. Providing active stock quotes, text from a book in a database, logic decisions based on user input, and a shopping cart are examples that require dynamic web pages.

### CGI Scripts

CGI scripts add to the functionally of the web server by allowing you to add custom functionality, like forms and dynamic HTML pages. Until recently, CGI was a Unix innovation. Under Unix, CGI scripts are not in the server and are executed from the file system. The server is set up so the execution environment for CGI scripts runs as standard output. But, in the AWC system, the CGI script runs as compiled code on the server and generates HTML, which is directed to the client browser over the network.

### How It All Connects

Figure 8 shows how a dynamic HTML web page is generated. The CGI code has buried ASCII text in the form of HTML lines, which are output when the code is run. Actions are performed using GCI standards that can be found in books like *CGI How-To*. These are used to develop parsing and other programming functions needed to act on the information transferred from the I/O drivers to the TCP/IP stack. When a box, verbiage, button, or other HTML function is needed, the code that provides the visual attribute is embedded in the CGI script.

The development of CGI improved web page programming. CGI is a mechanism by which you connect script to your web server. A CGI script causes the following five things to occur in a web server. First, the server creates a new session to execute the script. Second, a set of standard environment variables is set, including the remote host IP address and the URL that was specified. The script is then executed and the parameters are passed. Then, the web server captures any output generated by the script. And last, when the scripts end, the session is terminated and the captured output is sent to the browser.

CGI programs are script files that are written in a scripting language like Perl, or even C. Depending on the server, it may accept Visual Basic or batch files. The key is no user intervention is allowed during the execution.

I chose the CGI scripts to interface with the X-10 module to simplify the code and make it modifiable in the future. Application tools like Cold Fusion simplify writing CGI scripts. As virtual scoping advances, interfacing will be easier.

memory after the code is compiled. IP addresses, passwords, log on identification information, and the like are part of the RTOS. I took advantage of this storage location and used it for the house code for X-10.

This information is input using the console port (Port A) at powerup. The X-10 driver module also has two other inputs. The unit code selects the actual X-10 module to be addressed for control and the action for the X-10 module to execute. (You can download the software code of these functions from the *Circuit Cellar* web site.)

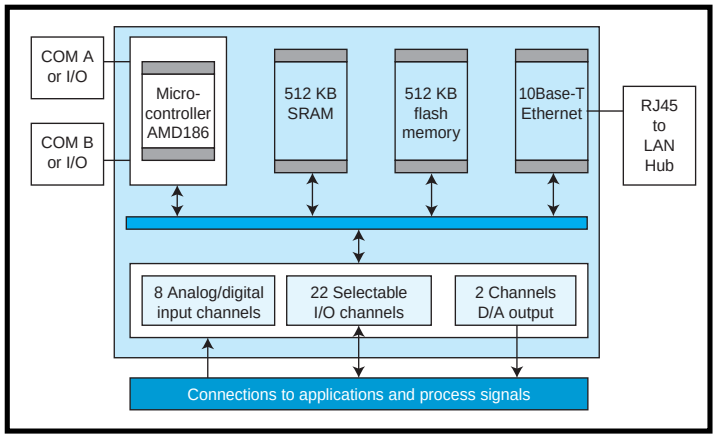


**Figure 2**—*The AWC86 block diagram provides some insight into the hardware functions of the Slim-Link.*

## FIRMWARE

You've had a basic tour of the hardware, now let's talk about the operating firmware. The flash memory provides a real-time task manager, dynamic web page host, and user web page CGI tools to support applications. Monitor firmware that uses serial port (COM A) is provided to support development and debug.

There is room left in the flash memory for the application firmware and web pages created for the user interface. Applications use the task manager to organize and simplify interface code development with the I/O lines and communication ports. The Slim-Link Server processor runs the server code and the application code at the same time.

## RTOS, THE SYSTEM BACKBONE

The AWC module uses MicroRTOS (XeRTOS), which allows real-time control of multifunction tasks. Product development moves smoother and quicker when tested in an RTOS. The AWC module also provides a useful multitasking capability. Multitasking allows the embedded web server to simultaneously provide more than the interface to X-10. Embedded applications always grow from the original concept, so expansion is useful.

The second phase of my selection process was finding a well-developed communication module. A TCP/IP communications protocol stack is provided in the AWC module for the server to communicate with the 'Net. The Ethernet device drivers and a server protocol were included. These are part of the RTOS in the Slim-Link Server. It is a full-featured software platform consisting of a preemptive real-time kernel, TCP/IP stack, and the AWC embedded web server.

Writing a driver is usually what an embedded programmer does best. The driver interface is the beginning of what web appliance programming is about. A dynamic HTML page generator needs to be connected to this driver. Flexible CGI seems to be the best solution for the C programmer.

People who are unfamiliar with C but understand script languages may prefer Perl, which is more simple. However, Perl is slower, because the server, must interpret, or compile, each time a page is generated. Perl has an interpreter that must reside in the server and AWC doesn't support it on the Slim-Link.

Setting up the AWC server involves entering the IP address using the console port. The settings allow

the embedded web server to provide an attachment to the network, which enables an external browser to communicate with the data in the server. The details and view of how to configure the web server are simplified with the AWC approach. Instead of searching for Ethernet card drivers and making them compatible with the operating system, the entire system has been configured, with the exception of the external Internet connection parameters. The Internet configuration is located in a well-documented file that requires minimal editing.

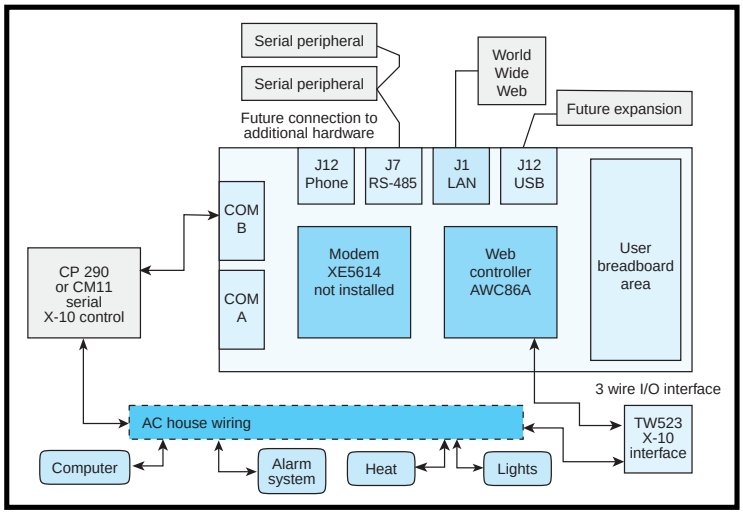


**Figure 3**—*Connecting the two X-10 interface modules is a rather simple process.*

The embedded web server interacts with the user application, the TCP/IP stack, and the real-time kernel. It accepts user application information, merges that information into dynamic web pages, and passes the pages to remote users using TCP/IP and network protocols. The real-time kernel includes a preemptive scheduler to handle multiple tasks running within the system.

The demo board comes preloaded with the AWC demo web site running on the web server. I replaced the application code with my X-10 application web pages. The AWC kit instructions guide you through bringing up the module and loading your application, HTML, and CGI codes. Because of the limited space in an embedded web server, the files are highly compressed; the maximum application code space is 64 KB. The building process AWC uses in the make files monitors the space. Figure 5 shows how to connect the server to load it and test the application code. I used the development kit's configuration because it only required a crossover cable to make the connection.

Listing 1 is an example of code from the AWC tool kit. It provides a button call to the file `Test.cgi`. This is the basic interface with the X-10 driver. The first important line, the control passing line, is in bold text. It furnishes the information to the server to execute the CGI code. The line begins with the form tag, which encloses all the elements in the form. The action attribute ensures the browser will receive the file.

You want the form generated by the CGI script returned to the browser. After "action", the CGI file name is given so the server can locate it. Then, "get" retrieves the file from the server. The typical directory location for an Apache server would be `/cgi-bin/Test.cgi`. The AWC RTOS simplifies things to only a root directory.

If a question mark was included after the file name, the server would pass everything after it to the CGI script. The plus between one and two clarifies spaces sent. Be careful with the characters. The slash should be represented as the hex %2F or it could be confused as a separator between directories. Rafe Colburn's *CGI Programming in a Week* discusses this in detail.

The name attribute sets a definition for this form so the server can identify it. In this case, it is

"memform." `Onsibmit=` tells the form to run the return `Check Entry ()` function when you press the submit button. This tests the input to make sure it's valid. When the server executes the CGI code, the output is piped back to the browser as HTML, which is presented as a new screen.

The second set of bold text in Listing 1 shows the submit button, which is an input type.

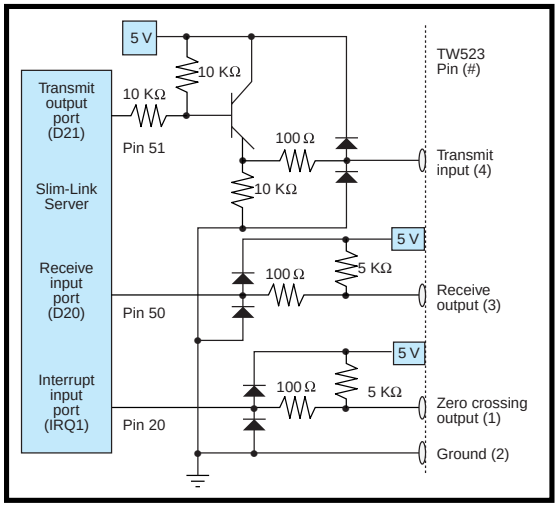


**Figure 4**—*According to the X-10 Powerhouse spec, this is what you'll need for components and values.*

## MicroRTOS PROVIDES TW523 INTERFACE

The MicroRTOS contains the firmware driver interface to the power line (for more information, read Donald Blake's article in *Circuit Cellar* 113). You would have to write and debug a time-consuming driver for the PIC processor, per X-10 specification. Choosing the Slim-Link module eliminated this development cycle.

The MicroRTOS driver uses precision background timers (1 ms and 1.778 ms) to measure the alternate phase zero crossings. The TW523 (Pin 1 with reference to ground Pin 2) gives a square wave representing the sinewave in the power outlet. This signal is sensed by the AWC module interrupt IRQ1 pin. The MicroRTOS activates the driver, which consists of an Interrupt Service Routine (ISR). The X-10 ISR is compiled in the code as part of a table. Although many ISRs are provided by the tool kit, the X-10 ISR fit my application.

Then, the RTOS assigns a timeslot for the ISR and begins processing it. The zero-crossing interrupt starts the 1-ms timer. When the timer elapses, it starts the 1.778-ms timer. Expiration of the latter indicates the first phase of a three-phase power grid. The 1-ms timer process is then restarted. A 6-bit register keeps track of the 1- and 1.778-ms timers.

Completion of the second set of time-outs indicates the second phase of the power. Next, the third phase is indicated by a counter value of six. Counter values are used to control the receiver and start the transmitter code accurately according to X-10 specification. Transmission of each bit takes 1 ms, which is driven by odd values of the counter. The silent time between each bit is 1.778 ms, which is the even bits of the counter. This background RTOS timer code allows the ISR in the MicroRTOS to handle the transmission and reception of X-10 information.

The CGI code passes the unit, house, and action codes to the driver (`xten.cgi` provided with tool kit) in MicroRTOS. It uses three 4-bit bytes that are the binary values of each code. They are combined into a 16-bit word. The `XECGIA01.c` module provided in the tool kit has the details of how to use this code when called by a CGI page. The command format for the `xten.cgi` driver module in MicroRTOS is seen in Table 1.

The CGI page that is calling the `xten.cgi` driver must preset the values for this word. The AWC example in file `XECGIA01.c` uses xt = 0x4020 to set up the house code to C, unit code to 1, and action to on. This value is placed in the MicroRTOS que and sent according to its timeslot. Software for the file `XECGIA01.C` is available for downloading.

## HTML CODE

The HTML and CGI code for the AWC web site is provided for you with the Slim-Link demo board. *Sams Teach Yourself CGI*, by Rafe Colburn, details the CGI language.

The HTML that represents the home page for the X-10 site (see Photo 1) may be downloaded. It supplies links to buy X-10 hardware. It also links to *Circuit Cellar*, where much of the early X-10 code for the 'x86 processors reside.

Listing 2 is the CGI code that makes this site function. It was originally written as HTML code using the Web Express Web Page 5.0 Construction Kit.

## HTML: FRIEND OR FOE?

HTML was not designed to align text, tables, or even images. As HTML became popular, it provided additional features. Despite advancement, I had trouble generating the house floor plan (see Figure 6) and placing icons for switches, controls, and sensors. Specific places relative to the floor plan just didn't exist and the screen moved with no predictable pattern. Alignment in HTML is not a simple issue. I compromised to make it work because absolute control is not possible with pure HTML.

Each platform acts differently when it comes to tables and forms for positioning. Microsoft and Netscape selected a different starting place for the upper left corner. I tested each concept in all the browsers before I was convinced my solution worked.

My first attempt was to place a floorplan in the background. But, it was impossible to control the position of items in the foreground. Next, I attempted to use tables. I used a 10 ×

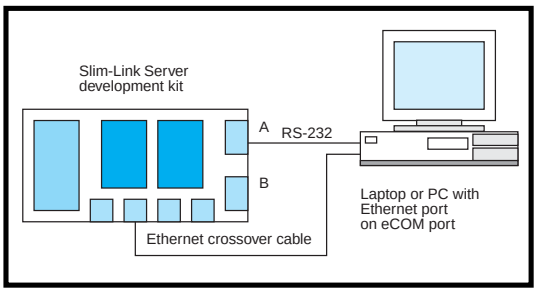


**Figure 5**—*This is all it takes to connect the Slim-Link for testing.*

15 table, which I assumed would position over the background exactly. However, the tables moved and changed size and font with each browser. The fonts set the size of the table cells and therefore lose relative position to the left edge. Linked words for the controls moved around the floor plan.

Pixel-transparent GIFs, which ensure better positioning, are a clever idea. But, working with pixels is monotonous. My next idea was to use tables by integrating the pixel system inside each cell. This requires cell building with pixels. Although slow, it's an improvement.

However, the background offset still plagued me. The final solution was to use frames and split the picture into two parts. A stationary right section shows the floor plan and a right frame shows the action of changing the X-10 modules.

I chose the dual frame window, and made the left frame the action area where all X-10 changes are made. The right frame remains static and the floor plan shows the location of the sensors.

Developing the floor plan is not difficult. A line draw program like Corel Draw or Word works best. The lines for the walls are placed on top of a set of temporary boxes. When the final lines are in place, the temporary boxes are removed allowing doorways and halls to appear. Then locations of switches are indicated with codes that refer to the buttons in the left frame. Save your work as GIF files. A notice will state the size is larger than the area open, but this is OK.

## SECURITY

The security of the floor plan page is critical. Having hackers come in the middle of the night to turn on your lights will produce the same feelings as a real burglar. Protection begins with the fact that this is an embedded web controller. The first level of security is the lack of a hard drive. After everything is converted to assembly code and stored in flash memory, changes can be made only from the console port, which isn't connected to the Internet.

The second security measure is a password. The MicroRTOS system allows you to change a password using the console port. To foil a hacker's attempt, I implemented a double password system (see Figure 7).

The double password system begins with entering the first password. The CGI page allows three tries, after which the CGI code requires a second password (reset password). The second password entered resets the CGI code enabling the first password to allow entrance into the floor plan page. The scheme works because a hacker doesn't know the password has changed. After the third erroneous entry, there is no difference in the "Password Error" screen.

If the reset password is entered, there are still only three chances to get into the floor plan page. If some-

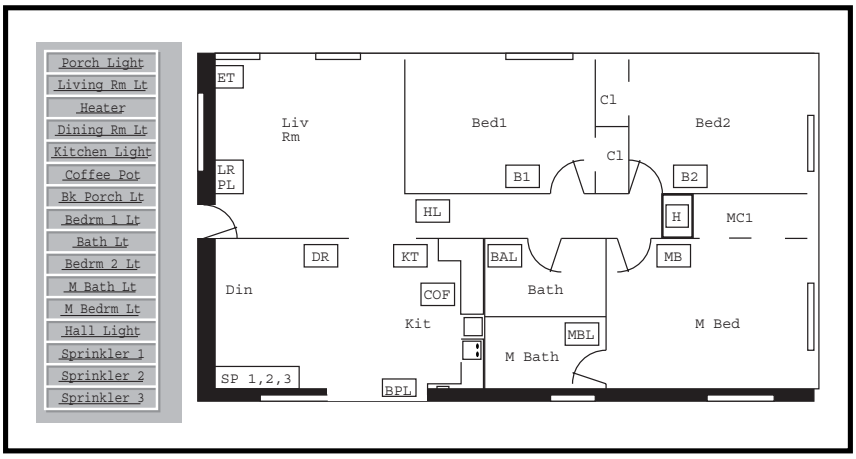


**Figure 6**—*Here's an overview of the floor plan, using a dual frame window.*

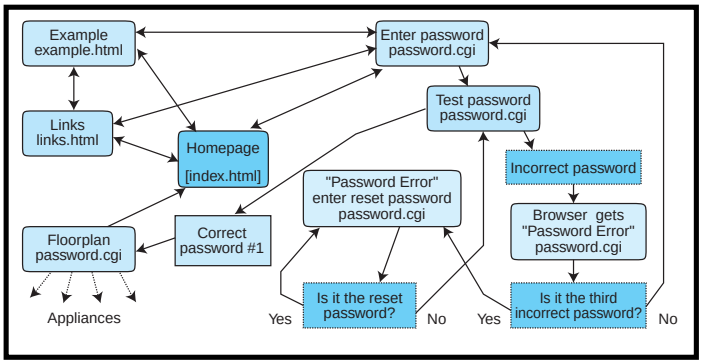


**Figure 7**—*With the dual password setup, your site is safer.*

one tries to hack in, you'll know. Your usual password will not work on the first try and you will need to use the reset password to clear the hacker's actions. After the reset password is entered, the correct password will work.

An improvement would be to encode or encrypt the password. Although unlikely, a hacker could read your password over the Internet. Public Key systems for 'x86 code are available on the 'Net and could be added as part of the CGI web page.

A final security feature is the link between the password and floor plan pages. If the floor plan page used a standard link from the password page, you could use a bookmark to get around the password.

To prevent this from happening, use a subroutine call from the password page to the floor plan page. This shows the web address for the password page when the browser is looking at the floor plan.

When the home page is selected from the floor plan page, it requires re-entry of the password. Listings 2 and 3 give basic guidelines for writing a CGI page that performs this. You need additional C code to provide the logic to incorporate the second password scheme.

## CGI CODE

Web servers started with static HTML web pages. Then, programmers began applying scripting languages. The output of the CGI script is HTML. Then, the server returns the script's HTML output to the browser (see Figure 8f). Writing server script in a more complex language than HTML allows logic to be added to the web page, creating a dynamic visual effect for the viewer.

A CGI script may have subroutines that interact with the browser and keep the same URL. This interaction allows one URL to have many functions—for example, turning on one of many lamps in the house. Creating a number of web pages with the lamps on and off in different combinations isn't practical. It's better to write a CGI script that makes a lamp a parameter and returns the correct on or off state.

## WRITING AN INTERACTIVE WEB PAGE

Writing HTML web pages isn't as exciting as embedded web pages. Visual scoping provides a new way to remotely see and change the world. Typically, embedded web pages are used to get information quickly. The worst mistake for an embedded web

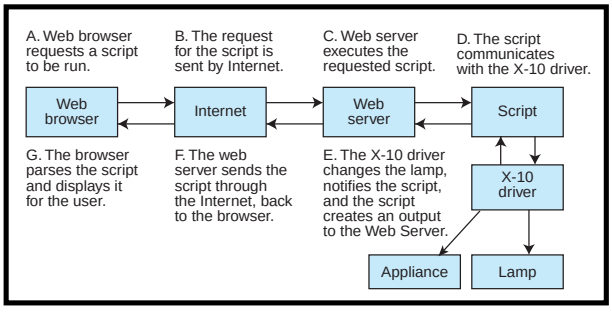


**Figure 8**—*Virtual scoping on the Internet is accomplished with a set pattern of steps.*

page is to overload it with unnecessary information. I keep my audience focused on the data being transferred. I make all supportable data in either HTML code or small visual images.

Complicated sites need an opening page like a star map to allow immediate one-click access. Figure 9 shows how any of the functions can be accessed with a single view.

When you plan your home scoping page, visualize how you'll access the controls. In Figure 9, the map for the site shows the index. A clever part of AWC's web pages uses framed sections that give guidance among pages, but do not have to be downloaded with each new page.

## GETTING STARTED

AWC provides many examples of both CGI script and HTML code with the Web-Controller Development Kit.

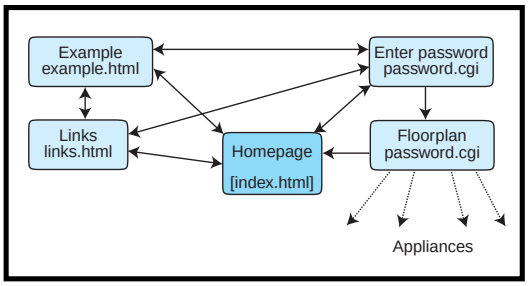


**Figure 9**—*Here's the homepage again. A block diagram is helpful in the planning stages to maintain a logical flow of the web site.*

The best way to get started is to do any development in real time. AWC offers the Slim-Link Server Evaluation Board Prototype kit that contains everything you'll need to get things up and running. The AWC89 is a good start if you don't want a modem. The AWC89A is the offering for developers who want a 56k auxiliary modem port.

When it comes to choosing a compiler, the Borland C version 3.0 or 4.5 is inexpensive and works well with the AMD186 processor. The old TASM assembler completes the tools to begin writing code.

I included several references at the end of the article that will help you. AWC provides further aid in identifying compatibility among hardware and compilers.

## INITIAL SETUP

Setting up the initial test and bringing up the Slim-Link Server module required two connections to my computer. There are several ways to connect to communications ports on the Slim-Link. The simplest way (see Figure 10) is used for development and testing. The Slim-Link LAN port is connected to a reversed Ethernet cable, which attaches to an Ethernet card in the PC.


**Photo 1**—*This is the homepage where I begin my virtual scoping.*

A second path from the PC is added by attaching one of the unused PC COM ports to the Slim-Link COM a serial port with an RS-232 cable. The house code needs to be set in each of the X-10 modules and each module's unit code must be coordinated with the floor plan page.

The first step is to test the AWC module. You can exercise the AWC module using a terminal program like Hyperterminal or PCPlus. The module's command line functions on the console port include PING, FTP client, e-mail, AWC Monitor, and Tele-monitor. You can enable or disable the web server functions from the console port, which is where you enter the passwords, IP address information, and X-10 house code.

After the console port information is entered, set up a browser window as a TCP/IP connection to access the Slim-Link Server's web LAN port. Follow the instructions in the demo kit for initial testing in demo mode before loading your compiled code through the console port. Then, you download your new program and begin testing it.

Debugging the embedded code is more difficult. A working web server and browser may be needed to emulate the code and web pages to make sure they work properly.

## TESTING

After testing, it was time to connect the AWC module to my real network. I have a simple firewall on the DSL line. Often, there is no firewall, yet the diagram is the same.

You can access the Slim-Link throughout the LAN within the firewall by using a fictitious IP assigned to it (see Figure 11a). A fictitious IP address would be common only to the LAN and not be registered on the Internet. Nobody beyond the firewall can access the Slim-Link's web server. All LAN participants can access FTP servers or hosts depending on the Internet connection. This step makes final testing easy. For final activation of the embedded server, a gateway is needed to connect to the outside world.

The diagram in Figure 11b shows how the web server accesses the world. Here, the server uses a real static IP address.

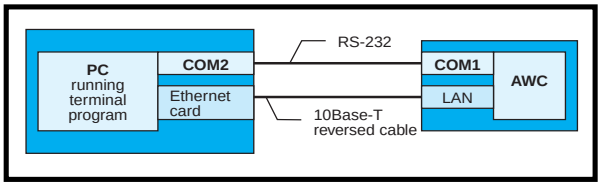


**Figure 10**—*Here's the basic connection scheme.*

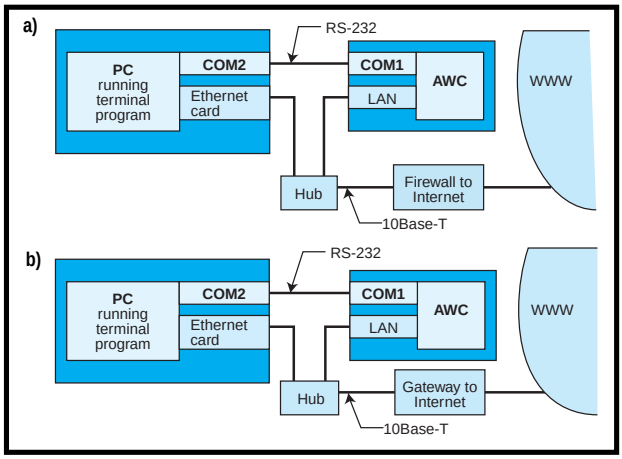


**Figure 11a**—*This is the typical test connection to the Internet.* ***b***—*And, here's the outside connection using a gateway.*

## APPLYING THE TECHNOLOGY

X-10 can give a freezer alarm or provide details of the home alarm system. Temperature control of the house, oven, water heater, and refrigerator can be monitored. With an additional file and voice application software/hardware, it can dial and play an emergency message to your cell phone. X-10 can monitor the moisture content of the lawn so you can remotely adjust the sprinkler timing. If any of your appliances have RS-232 service ports, Slim-Link could be used for remote access by technicians for diagnostics.

With this technology, geological stations can gather data about earthquakes, high water conditions, and so on. Heavy rainfall detected prior to flash flood conditions can be used to generate automatic warnings.

This article shows that using a module with a prewritten RTOS and driver saved months of toil. Programming is work, but it becomes fun when the code is simple.

*Dennis Wilkison works at Geometrics, a manufacturer of geological instruments. He has helped develop products that the first telephone credit verification terminal and first Ethernet switch. You may reach Dennis at wilkison@att.net.*

## SOURCES

**X-10**
X10 Inc.
(201) 784-9700
Fax: (201) 784-9464
www.x10.com

**Slim-Link Server**
Advanced Web Communications
A division of Xecom Inc.
(408) 945-6640
Fax: (408) 942-1346
www.xecom.com

**Compilers and software tools**
Borland C++ 3.0, 4.5/4.52 and Turbo Assembler
Microsoft MSVC++ 1.0/1.52
Microsoft FrontPage
Web Express WebPage Construction Kit 5.0

## REFERENCES

Rafe Coburn, *Sams Teach Yourself CGI*, Sams, Indianapolis, IN, 1998.

Fred Eady, "Embedded Internet Part 1: On the Network," *Circuit Cellar* 106, May, 1999.

Fred Eady, "Embedded Internet Part 2: On the Network," *Circuit Cellar* 107, June, 1999.

Ken Davidson, "Power-Line-Based Computer Control," *Circuit Cellar* 3, May/June, 1988.

Ken Davidson, "The X-10 TW523 Two-Way Power Line Interface," *Circuit Cellar* 5, September/October, 1988.

Don Blake, "A Temperature-Sensing Control Device," *Circuit Cellar* 113.

Warren Webb, "Embed the Web for Fun and Profit," EDN, March 18, 1999.

*Slim-Link Server Development Kit User Manual*, AWC, December, 1999.