FEATURE ARTICLE

Steve Freyder, David Helland, and Bruce Lightner

A $25 Web Server

"See the world's smallest web server. There are no age or height restrictions, but we're talking real excitement. Step right up and see how these guys put together a stand-alone web server that includes an Ethernet controller, real-time networking kernel, TCP/IP stack, and a PCB smaller than a business card.

The recipe is easy. Take an Atmel single-chip microprocessor, hook it up to an off-the-shelf PC network card, add a little code, and presto—you've got the web server shown in Photo 1. This circuit is so simple that you can build it in an evening and add worldwide web access to your favorite embedded application.

The PicoWeb server provides web access to digital I/O and serial I/O signals without the need for assistance from external PCs or Unix computers. It's a stand-alone device with a real-time networking kernel, a TCP/IP stack, and an HTTP web server. Plug the device into an Ethernet cable connected to the Internet, and you can control your sprinklers from any place on the planet.

This project started partly as an excuse to use a new microprocessor and partly to settle a long-standing argument about the possibility of delivering web pages with a commodity microcontroller. The Atmel AT90S8515 microprocessor looked exciting with its low-power RISC processor, 8 KB of flash program memory, 512 bytes of EEPROM, 512 bytes of RAM, 32 I/O lines, and a built-in UART. The realization that we could attach the Atmel part to an inexpensive PC ISA-bus network card with zero glue logic gave us a test vehicle to finally settle our argument (see Figure 1).

With an execution rate of one instruction per clock and a clock rate of 8 MHz, the AT90S8515 can transfer data over the ISA bus at full speed (about 2 MBps). All the hardware we needed to make a web server was quickly put into place. For debugging purposes, we hooked up the micro's serial port to a cable with an RS-232–level converter embedded in the DE-9 connector's hood. We also added a single LED for feedback.

After all that, we still had a little bit of money left in our $25 budget, so we threw in a $2 16-KB serial EEPROM chip to hold things like GIF and JPEG images (web pages need pictures, don't they?). With all the hardware together, it was just a matter of programming!

NETWORK ADAPTER

We chose an inexpensive "NE2000-compatible" PC Ethernet ISA-bus adapter for our breadboard setup. We needed to find an ISA-bus Ethernet adapter that could be configured to not use plug-and-play mode. Typically, disabling plug-and-play mode is done with an MS-DOS diagnostic program supplied with the network adapter.

True NE2000-compatible adapters will work out-of-the-box with the Novell/Anthem NE2000-compatible device driver supplied with Windows 95. Source code for MS-DOS packet drivers for NE2000 Ethernet adapters have long been available on the web, making such cards an excellent choice for projects like this.

We used an SN2000CT card that we picked up at our local Fry's Electronics store, but any true NE2000-compatible adapter will work if you can manage to turn off plug-and-play mode and set the adapter to a fixed I/O address. We

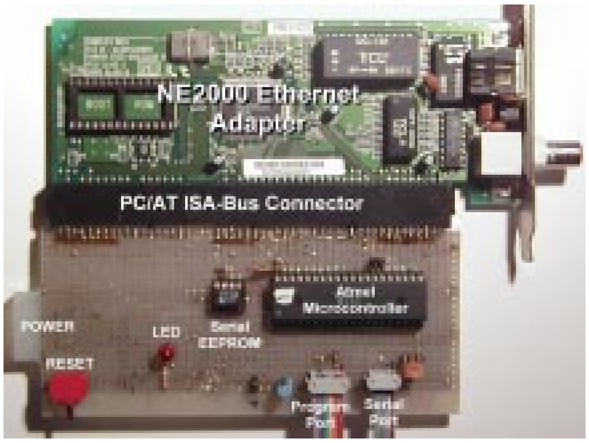


Photo 1—*Here's the PicoWeb server breadboard with a $9 ISA-bus network card and an Atmel AT90S8515 8-bit microcontroller. Connectors at the bottom provide a serial port and an in-circuit programming port using a PC parallel port cable. Power (150 mW typical) is supplied by a +5-VDC wall wart.*

chose an I/O base address of 0x300 for our project. Because we are not using interrupts, it doesn't matter how the card's interrupt request (IRQ) is configured.

Those familiar with the PC/AT ISA-bus will note that a 16-bit ISA card can use up to 88 unique bus signals, excluding power and ground connections (see Table 1). The Atmel microcontroller only has 32 I/O lines, so how do you connect a 16-bit PC adapter card to the micro?

First, the network adapter card isn't wired to all of the possible ISA-bus signals. We can easily determine which ones are used by looking at the copper traces coming from the circuit board's connectors.

Second, the network card doesn't need a number of other ISA-bus signals because of the mode in which we are using it (i.e., no DMA, no interrupts). And lastly, we can hardwire a number of the card's input signals (i.e., to +5V or ground).

Because we aren't using DMA or interrupts, we can ignore all of those signals, except AEN, which we wire to ground to indicate that DMA is not active. We need to hardwire *SMEMR to +5V to inhibit reads from the network card's optional onboard ROM and hardwire the upper 15 ISA-bus address bits (SA5–SA19) to match the I/O base address to which the adapter has been configured (i.e., 0x300).

This arrangement leaves us with only 25 signals that need to be wired up to the Atmel microcontroller—the 16 bidirectional ISA-bus data lines (SD0–SD16), the remaining five ISA-bus address lines (SA0–SA4), RESET DRV, *IOW, and *IOR. (By adding a latch for the re-maining five ISA-bus address bits, we could have shared those lines with the data bus signals, freeing up five more general-purpose I/O pins on the Atmel part.)

An NE2000-compatible Ethernet adapter is optimal because our processor is memory-limited. Such adapter cards have 16 KB of onboard SRAM—32× more than the Atmel microcontrol-ler's 512 bytes.

Part of the network adapter's SRAM functions as a ring buffer to allow unattended reception of back-to-back Ethernet frames in case the controlling processor is busy doing other things. The rest of the Ethernet controller's SRAM can be used to assemble transmitted Ethernet packets.

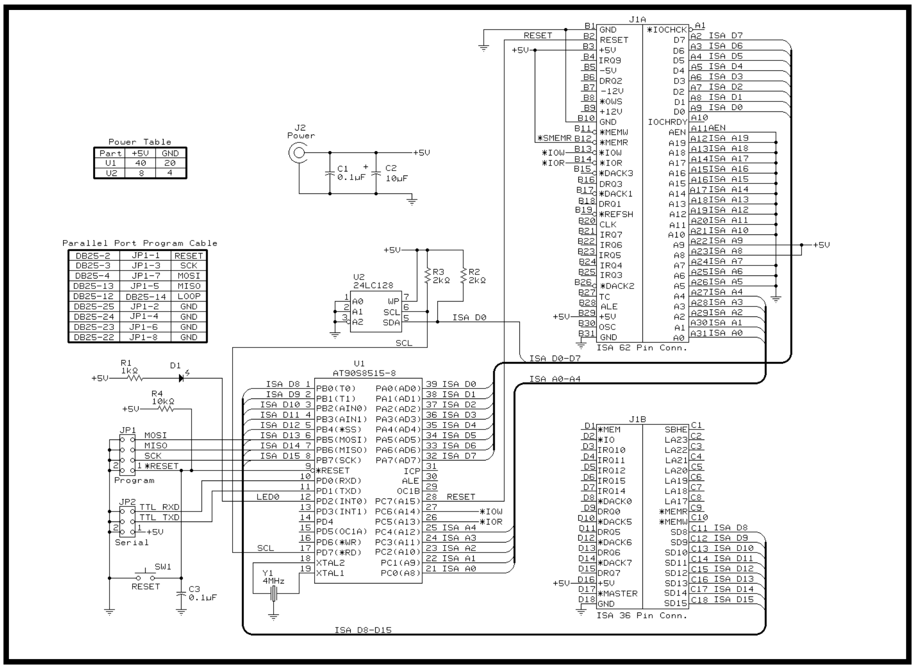
The net result is that very little of the Atmel microcontroller's meager 512 bytes of SRAM is needed for the sending and receiving of Ethernet packets. After all, a single Ethernet packet can be as large as 1500 bytes! The real clincher is that this board can be purchased for as little as $8.88.

FIRMWARE DESCRIPTION

The PicoWeb server's demonstration firmware supports a simple kernel, a tiny debug monitor, a p-code interpreter, a network adapter driver, a TCP/IP stack, and an HTTP server (i.e., web server). Let's look at what kinds of network messages the PicoWeb server responds to and what the responses are.

At the lowest level, the PicoWeb server responds to network ARP requests identifying it as an active device on the network with an assigned IP Address. Every PicoWeb server is assigned a unique Ethernet address that is sent as part of the ARP reply packet.

Next, we'll look at the BOOTP request. An IP address can be assigned to the device statically by storing the device's IP address in the microcontroller's flash memory or dynamically by using the BOOTP



Power Table

Part	+5V	GND
U1	40	20
U2	8	4

Parallel Port Program Cable

DB25-2	JP1-1	RESET
DB25-3	JP1-3	SCK
DB25-4	JP1-7	MOSI
DB25-13	JP1-5	MISO
DB25-12	DB25-14	LOOP
DB25-25	JP1-2	GND
DB25-24	JP1-4	GND
DB25-23	JP1-6	GND
DB25-22	JP1-8	GND

Figure 1—*The PicoWeb server breadboard makes a simple "glueless" connection of an Atmel AT90S8515 microcontroller to a PC ISA-bus connector. Also included is a 24C128 16-KB serial EEPROM, diagnostic LED, manual reset switch, connectors for power, serial port, and the in-circuit programming cable. The inset shows the wiring diagram for a PC parallel-port programming cable.*

protocol. When configured for dynamic IP address assignment, the PicoWeb server will begin sending periodic BOOTP requests after powerup until an appropriate reply is received.

The PicoWeb server's unique Ethernet address is broadcast as part of the BOOTP request packet. If a valid BOOTP response is received, the PicoWeb server uses the contents of the response to set its IP address. A Windows 95/98/NT version of a BOOTP server program is also available. This program uses a text file that maps PicoWeb server Ethernet addresses into assigned IP addresses.

The PicoWeb server also responds to ICMP echo, or ping, requests. The server responds to ICMP echo requests by responding with an echo reply. ICMP echo requests are typically generated by using a program called ping on a remote host. The ping program enables users to quickly test network connectivity with the PicoWeb server and evaluate the round trip time (RTT) of the echo request/replies.

The PicoWeb server can send and receive UDP packets. However, the breadboard prototype's demonstration firmware does not make use of this capability.

At the TCP/IP level, the PicoWeb server responds to HTTP GET requests that are addressed to its IP address. HTTP GET requests are sent by web browsers such as Netscape Communicator or Microsoft's Internet Explorer. Our web server responds to these requests by sending back HTML documents, text, images, and so on, just like a "real" web server. The only difference is that we don't need a giant OS with its attendant large memories, fancy TCP/IP stacks, expensive microprocessors, and high power consumption to get the same results!

The demonstration firmware purposely restricts the maximum size of an HTTP GET response to a single Ethernet packet (i.e., no more than 1400 bytes of TCP/IP payload) to conserve memory resources. In the context of an embedded web server using this class of low-cost microcontroller, this restriction is not an unreasonable tradeoff. A number of HTML coding techniques can be used to work within these limits, including the use of HTML frames and the "gluing together" of multiple GIF and JPEG images using things like HTML tables.

The demonstration firmware's basic response to an HTTP GET request is shown in Photo 2. The basic response is to return a web page that shows a title, two radio buttons, an update button, and a few JPEG images. The two radio buttons show the state of the LED on the demonstration board.

The user can click the radio buttons to change the state of the LEDs (i.e., to on or off) and then click the Set LED button to send the new state information to the PicoWeb server. The server responds by setting the LEDs according to the request and then updates the returned web page to reflect the current state of the LEDs. Those of you familiar with server-side web programming will recognize this as the typical behavior of a cgi-bin script. The images shown on the sample web page are also supplied by the PicoWeb server.

The PicoWeb server demonstration firmware contains a simple, extensible debugger that provides for things like memory dumps, EEPROM alteration, p-code and network tracing control, and more. Debugger commands such as those in Table 2 can be entered via the serial port, or via the network using a web browser and a URL that refer-

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A1	I/O CH CK	B1	GND	C1	SBHE	D1	*MEM CS16
A2	SD7	B2	RESET DRV	C2	LA23	D2	*I/O CS16
A3	SD6	B3	+5 VDC	C3	LA22	D3	IRQ10
A4	SD5	B4	IRQ2/9	C4	LA21	D4	IRQ11
A5	SD4	B5	–5 VDC	C5	LA20	D5	IRQ12
A6	SD3	B6	DRQ2	C6	LA19	D6	IRQ15
A7	SD2	B7	–12 VDC	C7	LA18	D7	IRQ14
A8	SD1	B8	0WS	C8	LA17	D8	*DACK0
A9	SD0	B9	+12 VDC	C9	*MEMR	D9	DRQ0
A10	I/O CH RDY	B10	GND	C10	*MEMW	D10	*DACK5
A11	AEN	B11	*SMEMW	C11	SD8	D11	DRQ5
A12	SA19	B12	*SMEMR	C12	SD9	D12	*DACK6
A13	SA18	B13	*IOW	C13	SD10	D13	DRQ6
A14	SA17	B14	*IOR	C14	SD11	D14	*DACK7
A15	SA16	B15	*DACK3	C15	SD12	D15	DRQ7
A16	SA15	B16	DRQ3	C16	SD13	D16	+5 VDC
A17	SA14	B17	*DACK1	C17	SD14	D17	MASTER
A18	SA13	B18	DRQ1	C18	SD15	D18	GND
A19	SA12	B19	*Refresh				
A20	SA11	B20	CLK				
A21	SA10	B21	IRQ7				
A22	SA9	B22	IRQ6				
A23	SA8	B23	IRQ5				
A24	SA7	B24	IRQ4				
A25	SA6	B25	IRQ3				
A26	SA5	B26	*DACK2				
A27	SA4	B27	T/C				
A28	SA3	B28	BALE				
A29	SA2	B29	+5 VDC				
A30	SA1	B30	OSC				
A31	SA0	B31	GND				

Table 1—*The PC/AT ISA-bus signals in red and green are required by the NE2000 network adapter. The signals in green are connected to the Atmel microcontroller and the signals in red are "hardwired" to power and ground. We were able to connect the Atmel micro to the ISA bus with no extra logic.*

ences a special TCP port (i.e., port 911).

The format of a debugger command URL is `http://IPaddress:911/command[[+parameter1]]+paramet-er2]`

Any results from executing a debug command will be returned as a web page. For example, *`http://IPaddress:911/dm+60+80`* will list the contents of the first 128 bytes of the microcontroller's SRAM. New debugger commands can easily be added (or deleted to save program code space).

The PicoWeb server's prime function is to return web pages and images in response to HTTP GET requests directed to URLs targeting its HTTP server. The PicoWeb server's firmware responds to URL queries directed to TCP port 80 in a conventional manner. Because the PicoWeb server doesn't have a true file system, URLs trigger dedicated routines in the firmware as opposed to simply returning the contents of a disk file.

A summary of the standard URLs implemented in the PicoWeb server's demonstration firmware are shown in Table 3 (note that the *`http://IPaddress`* part of the URL does not appear in the table). Either *`http://IPaddress`* or *`http://IPaddress/ii00.html`* will retrieve the default web page (home page) from the PicoWeb server.

SOFTWARE DEVELOPMENT

You're probably wondering: "How does all of this fit in an 8-KB microprocessor?" The answer: "Very carefully!" We wrote the software to efficiently implement the necessary network protocol layers by using a p-code technique to conserve code space in exchange for somewhat reduced execution speed.

Time-critical network code is left in native Atmel RISC code. As a result, the actual space used by the PicoWeb server demo firmware is under 7000 bytes, including debugging code. Therefore, space remains for developers to roll their own code and add even more functionality to

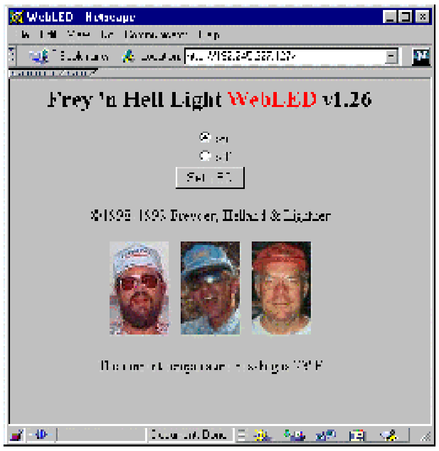


Photo 2—*Everything on this web page is delivered directly by the PicoWeb server including the JPEG images. A user can change the state of the PicoWeb's diagnostic LED using the radio buttons. The temperature shown is from a Dallas DS1621 two-wire digital thermometer chip attached to the PicoWeb using a couple of unused digital I/O lines.*

the PicoWeb server.

Because p-code can be run from the serial EEPROM chip (at a greatly reduced execution rate), a substantial amount of added functionality is possible. The present development environment allows custom application development without the need for access to the underlying real-time networking kernel source code. The source code for the PicoWeb server real-time networking kernel is available for license by serious developers.

The software development environment used for the project made use of the Atmel free assembler, which can be downloaded from Atmel's web site.

To enhance the capabilities of the assembler, a Windows version of the GNU C preprocessor was used to add certain macro/include file capabilities that were conspicuously missing from the Atmel product. Atmel sells a $49 development kit for the AT90S8515 that enables users to quickly get up to speed and download programs into the AT90S8515's flash memory.

The AT90S8515 processor allows in-circuit programming of its flash memory via a four-wire SPI interface. To eliminate the need for the $49 development kit, a C program was written to enable a PC to program the Atmel microprocessor on our breadboard in-circuit via a cable attached to a PC parallel port.

The HTML-like code in Listing 1 displays a web page that provides the status of the breadboard's onboard LED and provides an HTML form that enables the user to control the LED.

Special tags beginning with a back-tick (`) are embedded in standard HTML code. These tags invoke firmware routines when a web page containing them is returned to the requestor. The tags can be used to dynamically insert variable data and text into a web page when ref-erenced. Table 4 shows a number of the tags implemented in the PicoWeb server demonstration firmware.

The tag `` `t `` outputs a standard HTML `text header`. The tag `` `100 `` outputs the string `input type=` and is used as an example of how to save EEPROM storage space. `` `001 `` is an example of a dynamic tag. It is part of a kind of "if-then-else" construct. In Listing 1, if the value of output bit 1 is zero (i.e., the LED is on), the first radio button on the web page will include the `CHECKED` string; otherwise the string is omitted. If the output bit 1 is one (i.e., LED is off), then the second ratio button will include the `CHECKED`

Command	Description
dm XXXX nn	dump SRAM from XXXX...XXXX+nn-1
de XXXX nn	dump EEPROM from XXXX...XXXX+nn-1
ds XXXX nn	dump serial EEPROM from XXXX...XXXX+nn-1
wm XXXX YY	write SRAM address XXXX with byte YY
we XXXX YY	write serial EEPROM address XXXX with byte YY
ws XXXX YY	write EEPROM address XXXX with byte YY
l	toggle TCP packet logging on/off
pd n	control p-code debug trace (0=off; 1=on)
PC XXXX	call p-code routine at address XXXX
R	reset processor
^C	reset processor (serial port only)

Table 2—*Debugger commands like these can be sent over the network using a web browser or via the PicoWeb's serial port.*

URL	Description
/	Return document ii00.
/ii*hh*	Return document number *hh* to user. Documents numbers are two-digit hex values. Anything after *hh* in the URL is ignored.
/iuhh	Call firmware routine number hh. Mostly useful for testing "html include" routines.
/x?*n*=*value*	Set digital I/O port bit *n* to *value* where *value* is either 0 or 1. Because the breadboard LED's anode is connected to bit 0, the command "/x?2=0" turns on the LED.

Table 3—*Certain PicoWeb server firmware routines can be activated by referencing special URLs. This arrangement provides simple remote control of the Atmel micro's digital I/O lines.*

string.

An example of how to extend the functionality of the PicoWeb server is shown at the end of Listing 1. In this case we hooked up a Dallas DS1621 two-wire digital thermometer chip to our breadboard. The tag `` `701 `` invokes a p-code routine stored in serial EEPROM that takes a reading from the Dallas chip and returns a text string with the decimal temperature reading converted to degrees Fahrenheit. The resulting real-time temperature can be seen in Photo 2.

Another way to add features to the returned web page is by using the serial port to talk to an external device that supports serial communications (e.g., a test instrument). A p-code routine triggered by a special tag can send a command out the serial port to the external device. The external device can then interpret the command and send data back to the PicoWeb server over the serial port. That data can be inserted directly into the returned HTML web page or additional p-code can be executed to format the data before insertion in the returned web page.

Note in Listing 1 that three JPEG images (`ii01.jpg, ii02.jpg,` and `ii02.jpg`) are referenced by the demo web page. These JPEG images are supplied by the PicoWeb server from the breadboard's serial EEPROM.

FIRMWARE DEVELOPMENT ENVIRONMENT

A p-code instruction interpreter was developed for the PicoWeb server. The use of p-code provided program code simplification, the option to execute program code out of EEPROM (including external serial EEPROM), and in many cases, a reduction in program code size when compared to native code.

Code simplification is achieved because the p-code makes no reference to native registers, and because the p-code "virtual machine" uses 16-bit-wide data types for most operands. P-code execution from EEPROM is possible because the p-code instruction pointer is 16 bits wide, with the uppermost bit indicating (when set) that the next p-code instruction should be fetched from EEPROM and not from the Atmel microcontroller's 8-KB flash program memory. Program size reduction results from a combination of factors, including efficient application-specific p-code routines and flexible p-code operand addressing modes.

A description of the many p-code routines is beyond the scope of this article, but user p-code documentation is supplied with the breadboard software development package.

The software build environment is based on Windows 95, 98, or NT. Although we use the standard Atmel assembler for firmware code generation, the source code is first passed through a public-domain C preprocessor and a Perl script before being sent to the assembler. The build procedure is controlled by a simple batch file.

The build batch file performs a series of steps, the first of which is to run the C preprocessor to generate an intermediate file (`.i`) from a set of input source (`.asm`) files. The next step is running a Perl-based string extraction program that collects all of the quoted text strings and stores them in a separate file for inclusion (using `.include`) at the end of program memory.

As the third step, the batch file runs a Perl script that spawns the assembler and postprocesses any error messages to convert the associated line number back to an original input file name and line number. The postprocessed error output is written to standard output and is compatible with the error browser provided by Microsoft's C++ Visual Studio.

Next, a Perl script that post-processes the `.EEP` file to separate the Atmel-based EEPROM segments from the external serial EEPROM segments is run.

The final step is to run a Perl

```
`t<html>
<head><title>WebLED</title></head>
<body text=#000000 bgcolor=#c0c0c0>
<center>
<h2>Frey 'n Hell Light <font color="red">WebLED</font> v1.26</h2>
<FORM name=mfrm method=GET action="/x">
<`100radio NAME=4 VALUE=0 `001CHECKED{}>on<br>
<`100radio NAME=4 VALUE=1 `001{CHECKED}>off<br>
<`100submit VALUE="Set LED">
</FORM>
&#169;1998-1999 Freyder, Helland & Lightner
<br><br>
<img src="ii03.jpg" width=64 height=100 alt="Frey">&nbsp&nbsp
<img src="ii02.jpg" width=64 height=100 alt="Hell">&nbsp&nbsp
<img src="ii01.jpg" width=64 height=100 alt="Light">
<br><br>
The current temperature reading is `701&#176;F
</center>
```

Listing 1—*This HTML-like code delivered the web page shown in Photo 2. The special tags beginning with a back-tick (`) activate PicoWeb firmware routines that insert text into the HTML document stream when the page is retrieved by a web browser.*

Tag	Meaning
`t	Emit HTML header string
`0hh	If port bit *hh* is low; emit HTML text from stream up to '{', then skip text up to next '}'. If port bit *hh* high,skip HTML text up to next'{', then emit text up to next '}'.
`1hh	Emit string number *hh*.
`7hh	Invoke "user p-code" routine number *hh*.

Table 4—*Special tags embedded in PicoWeb web pages invoke firmware routines that can insert text into a dynamic HTML code stream. These tags can be used to do things like take a temperature reading and return a text string with the current temperature.*

script that processes a text file with a list of web pages and images to be included in the build and produces a binary load image file suitable for use with our network-based serial EEPROM loader (`netprog.pl`). The complete build procedure only takes a couple of seconds on a Pentium II–based PC.

A full software build produces the following files that need to be downloaded into the PicoWeb server:

- `picweb.rom`—Atmel flash code/data in generic ROM format
- `picweb.ep`—Atmel EEPROM data in generic ROM format
- `picweb.el`—serial EEPROM p-code/data in EEPROM loader format
- `images.dat`—serial EEPROM HTML, GIF, and JPEG data

The first two files are used to program the flash memory in the Atmel microcontroller, and the last two files are used to program the PicoWeb server's onboard serial EEPROM.

Programming of all flash memory and EEPROM data is controlled via a batch file that invokes the programming utility to program the microcontroller's flash memory with the contents of `picweb.rom` and (if nonempty) programs the on-chip EEPROM with the contents of `picweb.ep`.

The batch file also invokes the Perl-based `netprog.pl` script to program the serial EEPROM with the contents of `picweb.el` and `images.dat` (via the network).

Programming the Atmel microcontroller takes less than 20 s. Programming the serial EEPROM via the network connection takes a similar amount of time, depending on the amount of p-code and web page data placed in the serial EEPROM.

We developed the parallel port programming tool (PPPT) so that it would not be necessary to remove the Atmel microcontroller chip from the breadboard and plug it into a separate programmer to change the firmware. Our program was inspired by the BASIC program from Jeff Bachiochi's "Learning to Fly with Atmel's AVR" article (*Circuit Cellar* 101).

The PPPT is a 16-bit MS-DOS program written in C that will run under Microsoft Windows in an MS-DOS Prompt window. The program requires a simple cable that you can build yourself (see Figure 1 for the cable pinouts). By plugging the cable into the programming connector on the PicoWeb server, the AT90S8515 can be programmed in-circuit using a PC's parallel port.

We like our program better than the similar program provided with the Atmel AVR demo kit because, besides being free, PPPT reprograms Atmel parts about twice as fast as the AVR demo kit. PPPT also has both command-line and menu-driven interfaces (see Table 2 for a summary of the commands). Here's a sample command line we use (in a batch file) to erase, download both program memory and EEPROM, and then reset the PicoWeb server:

```
pppt -ce -lp test.rom -le test.eep -en
```

Our programming tool allows the AT90S8515's program memory and EEPROM to be dumped, modified, saved, or loaded via standard `.ROM` files produced by the Atmel assembler.

Menu command	Command description
CE	Chip Erase
WL	Write Lock bits
LP	Load Program memory from file
LE	Load EEPROM from file
SP	Save Program memory to file
SE	Save EEPROM to file
DP	Display Program memory
DE	Display EEPROM
DD	Display Device codes
AP	Alter Program memory
AE	Alter EEPROM
Q	Quit program

Table 5—*Our PPPT can be used to quickly download the PicoWeb micro's program and data memory in-circuit using a standard PC parallel port. A number of PPPT commands are available to load, dump, and patch the Atmel chip's flash memory via its simple four-wire SPI programming interface.*

THE NEXT GENERATION AND BEYOND

Eventually we got the PicoWeb server breadboard working and settled our argument about the

Photo 3—*Here's the fully integrated PicoWeb server.*

feasibility of putting cheap microcontrollers on the Internet. We proved that cheap web-enabled embedded microcontrollers are indeed feasible, but our breadboard wasn't pretty and it was way too big!

The next step was to take the PicoWeb server breadboard's components, combine them with the ISA-bus Ethernet adapter's components, and lay everything out on a single PCB. To save on circuit board area, the surface-mount version of the Atmel AT90S8515 was used along with a low-cost highly integrated version of an NE2000-compatible Ethernet controller, the Realtek RTL8019AS. The result is a simple two-sided 1.4″ × 3″ circuit board (smaller than a business card) that fits into an extended DB-25 connector hood (see Photo 3).

The Realtek RTL8019AS Ethernet controller is a single-chip NE2000-compatible device with on-chip RAM that only needs a transformer, a single resistor, and a few capacitors to implement a complete 10BaseT Ethernet network connection. Therefore, an Atmel AT90S-8515 microcontroller, a Realtek RTL8019AS Ethernet chip, two crystals, a transformer, RJ-45 and DB-25 connectors, and a few resistors, capacitors, and LEDs are the only components needed to construct this ultra-small PicoWeb server.

While we were at it, we added a +5-V regulator and a Maxim RS-232–level shifter. We kept the breadboard's 16-KB serial EEPROM chip. Because the Realtek chip supports an 8-bit data bus, we were able to free up eight more I/O pins on the Atmel microcontroller. As a result, the DB-25 connector has 16 free digital I/O lines, as well as an RS-232 serial port. The total parts cost remained under our $25 target.

The printed-circuit version of the PicoWeb server supports all the functions of the breadboard version plus a few additional features. There are now 16 bits of external digital I/O available over the DB-25 connector. Two of the DB-25 pins carry the AT90S8515's serial port transmit and receive lines using standard RS-232 levels. In-circuit programming of the Atmel microcontroller also is via the DB-25 connector.

The onboard regulator accepts either AC or DC power in the range of 6–30 V. Typical current consumption is under 30 mA from the +5-VDC supply.

What we seem to have created is the world's smallest, cheapest, and lowest power web server. However, given the rapid pace of technology, these records surely won't stand for long!

Steve Freyder telecommutes from home for Science Applications International Corp. (SAIC) working on automated toll-collection systems. He lost his office at SAIC in San Diego many years ago by never visiting it. Steve has been programming since he first discovered computers in high school in 1970. You may reach him at steve@freyder.net.

David Helland works for SAIC, most recently on portable electronics for military training range systems. Dave has been building hardware and software systems for several decades now, and in his spare time, he restores vintage fiberglass dune buggies. You may reach him at dhelland@worldnet.att.net.

Bruce Lightner works from home for Lightner Engineering in La Jolla, CA. He too discovered computers several decades ago and has been building hardware and software for them ever since. You may reach him at lightner@lightner.net.

Collectively, the authors are often referred to as "Frey n' Hell Light" by their friends.

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199 or subscribe@circuitcellar.com

SOFTWARE

Software for the PicoWeb server breadboard is available via the Circuit Cellar web site.

REFERENCES

Atmel Corp., "AT90S4414/AT90S8515—8-bit AVR Microcontroller with 4K/8K Bytes In-System Programmable Flash," Rev. 0841E–04/99, 1999.

RESOURCE

Information on the printed-circuit version as well as software, firmware, and development tools for the PicoWeb server can be found at www.picoweb.net/.

SOURCES

AT90S8515
Atmel
(408) 441-0311
Fax: (408) 436-4200
www.atmel.com

RTL8019AS
Realtek Semiconductor Corp.
+886-3-5780211
Fax: +886-3-5776047
www.realtek.com.tw

DS1621
Dallas Semiconductor Corp.
(972) 371-4000
Fax: (972) 371-3715
www.dalsemi.com

NE2000-Compatible Ethernet Adapters
Allied Telesyn Int'l.
(800) 424-4284
Fax: (425) 489-9191
www.allied-telesyn.com

PicoWeb server
Lightner Engineering
(619) 551-4011
Fax: (619) 551-0777
www.picoweb.net