

## PicoWeb “WebLED” Project

This PicoWeb project can be found in the Picoweb development system “samples” directory in a file named `samples\webled\webled.pwp`. This project consists of a single HTML document or Web page (`webled.htm`) which references three JPG images (`bruce.jpg`, `dave.jpg` and `steve.jpg`). This project allows a PicoWeb server to display the “home page” shown in Figure 1.

A single user-supplied application-specific CGI routine, written in PicoWeb pcode, is also required to support this simple PicoWeb project. This CGI routine is used to read the current temperature from an I<sup>2</sup>C digital thermometer chip attached to the PicoWeb server’s 25-pin I/O expansion connector. (You can still build and load this project’s firmware into a PicoWeb server without an attached digital thermometer. If the I<sup>2</sup>C chip is missing, the PicoWeb’s firmware will always read -1 degrees Celsius from the missing chip.)

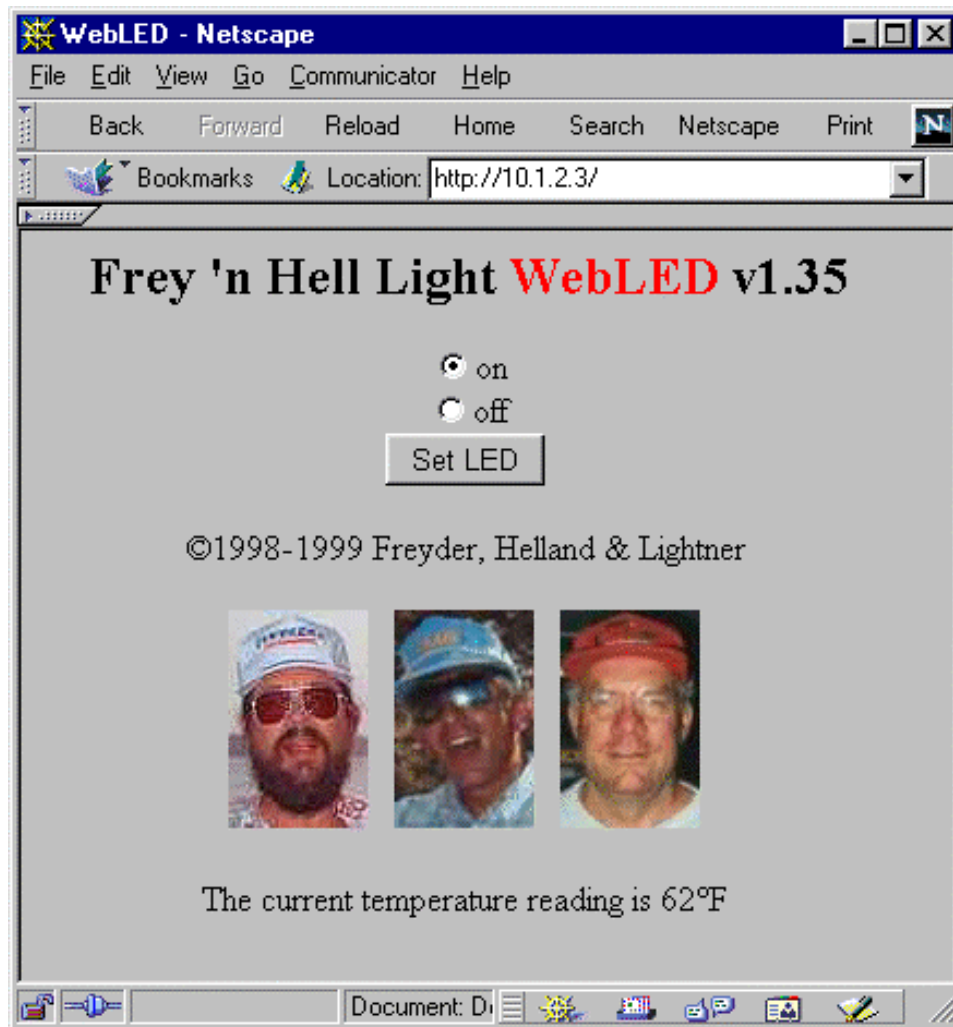


Figure 1- PicoWeb WebLED Project Web Page

**Project File** - The various parts of the WebLED project file (webled.pwp) will be described individually in the following sections:

```
//-----  
//  
// PicoWeb Project File for WebLED  
//  
//-----  
//  
// application-specific preprocessor definitions  
//  
#define BANNER "\r\nPicoWeb WebLED\r\n"  
#define EEPROM_IP /* use file "ip" for IP address */  
#define ENABLE_WATCHDOG /* use Atmel wathcdog timer hardware */  
#define DEBUGGER /* include debugger firmware */  
#define SERIAL_BAUD_DIVISOR 25 /* serial port speed: 19200 baud @ 8 MHz */  
  
#define DS1621_BASE_ADDR 0x9E /* ext. DS1621 digital thermometer */  
#define I2C_PORT1_PRESENT /* enable I2C code for DS1621 */
```

The project file webled.pwp also contains several preprocessor #define statements, as follows:

- BANNER – specifies a text string that is printed each time the PicoWeb server is reset
- EEPROM\_IP – specifies that the PicoWeb IP address is stored in on-chip EEPROM (the address will come from the file ip).
- ENABLE\_WATCHDOG – enables the watchdog timer hardware on reset
- DEBUGGER – includes the serial port debugger firmware
- SERIAL\_BAUD\_DIVISOR – sets the baud rate for the serial port (used by the debugger)
- DS1621\_BASE\_ADDR – defines the base address of external I<sup>2</sup>C digital thermometer chip.
- I2C\_PORT1\_PRESENT – enables firmware that accesses the external I<sup>2</sup>C digital thermometer chip

All but the last two defines are normally required as part of any PicoWeb project file.

```
//  
// application-specific HTML and image file names  
//  
webled.htm // ii00 (default Web page)  
bruce.jpg // ii01  
dave.jpg // ii02  
steve.jpg // ii03
```

This section of the project file lists the files which contain HTML code and images which will be stored in the PicoWeb serial EEPROM memory as part of the Web server's "file system". The first file in the list must be the default Web server "home page". Everything after the //s is comment text, and is ignored by the software which processes the project file. The text of the form *iihh* after each file indicates the URL "file name" that will be assigned to each file after it is placed in the PicoWeb server's file system.

```
//  
// application-specific CGI routines  
//  
temperature.cgi // iu00 (returns ASCII temperature reading (deg-f))
```

The above line indicates that a user-supplied CGI pcode routine named "temperature" needs to be loaded into the PicoWeb server. Also, wherever the string `temperature.cgi` is found in any of the HTML source code files loaded into the PicoWeb server file system as part of this project, a special *tag* will be inserted into those files. This special tag will cause the pcode routine "temperature" to be called at that point in the HTML code stream whenever a HTML file with that tag is retrieved.

```

#avr_reset
;-----
; this code is executed each time microcontroller is reset
;-----
;
    sbi        ddrd,LED_BIT        ; make LED driver pin an output

    ldi        yl,0xEE             ; start DS1621 temperature conversion
    rcall     dsl621_write

```

The above AVR assembly language will be inserted into the PicoWeb firmware build such that that code is executed each time the Atmel microcontroller is reset. The first line of code makes the microcontroller pin which drives the single user-controlled LED on the PicoWeb server an output pin (i.e., pin PD4). The next two assembly language instructions initialize the I<sup>2</sup>C digital thermometer chip.

```

#avr_slow
;-----
; this code is executed each trip through "slow idle" loop (~1 sec period)
;-----
;

#avr_fast
;-----
; this code is executed each trip through "fast idle" loop
;-----
;

```

The #avr\_slow and #avr\_fast code sections are empty. In fact, those directives could be removed from the project file. These sections allow the user to insert code in the inner (fast) and outer (slower) idle loops in the PicoWeb server firmware. The “fast idle loop” is where the PicoWeb server firmware polls the Ethernet controller chip. The outer “slow idle loop” is executed about once per second, and is used to check on less demanding tasks.

```

#avr_asm
;-----
; application-specific CGI pcode and AVR assembly routines go here
;-----
;

;-----
;+
; **--temperature-show the temperature (deg-F) in decimal.
;-
;-----
.eseg
temperature:
    pcall tempf_buf                ; read temp in deg-F
    pcall deconv                   ; convert to decimal ascii
    pret

```

The #avr\_asm project file directive specifies the beginning of user-supplied, application-specific routines which do not need to be executed at reset time, nor every time through one of the “idle loops”. This is where user-supplied CGI routines are placed.

The above pcode section is a PicoWeb pcode routine named `temperature` which, when called, will read the current temperature from the digital thermometer chip and output that reading in ASCII text to the *standard output stream*. If this routine is called because an HTML page is being retrieved which contains a special tag pointing to this routine, the *standard output stream* will be pointed at the HTML code stream. Therefore the latest ASCII temperature reading will be automatically inserted into the HTML code stream in place of this routine's tag. Note the `.eseg` directive which causes this code to reside in the PicoWeb server's external serial EEPROM memory.

```

;-----
;+
; **-tempf_buf-read temp (deg-C) from DS1621 and convert to deg-F
;
; inputs:
;   none
;
; outputs:
;   buf = latest temperature reading in degrees Fahrenheit
;-
;-----
.cseg
tempf_buf:
    pclrw buf
    pread_temp buf                ; get temp
    pmovwi buf+4,0x80             ; set up for sign test
    pandwi buf+4,[buf]           ; check the sign
    pjumpeq temp_positive        ; it is positive
    pmovbi buf+1,0xff            ; extend the sign
    pnegw buf                     ; negate
temp_positive:
    pmul buf,[buf],9              ; C*9
    pdiv buf,[buf],5              ; /5
    pandwi buf+4,0x80            ; test original sign
    pjumpeq temp_norenegate      ; positive
    pnegw buf
temp_norenegate:
    paddwi buf,32                  ; +32 - buf now has temp (F)
    pret

```

This PicoWeb pcode routine reads the current temperature from the digital thermometer chip then converts that reading from degrees Celsius to degrees Fahrenheit. Temperature is read from the chip as a single signed 8-bit value in degrees Celsius. After multiplying that reading by 9, dividing by 5, and adding 32, the resultant 16-bit value is passed back to the caller in the SRAM location `buf`. Note the `.cseg` directive which causes this routine (and those which follows it) to reside in the PicoWeb server's internal program memory.

```

;-----
;+
; **-deconv-decimal converter.
;
; inputs:
;   buf = word to convert
;
; outputs:
;   result printed (with pputc)
;-
;-----
#define DIG_BUF buf+6
#define DIG_PTR buf+4
#define DEC_REM buf+2
#define DEC_VAL buf

.cseg

```



The special tag ``temperature.cgi`` was discussed previously. It tells the PicoWeb server to call the user-supplied pcode routine “temperature” at that point in the HTML code stream where that tag is located.

The following lines in Listing 1:

```
<input type=radio NAME=4 VALUE=0 `?04 CHECKED{}>on<br>
<input type=radio NAME=4 VALUE=1 `?04{CHECKED}>off<br>
```

need to be explained. A tag of the form:

```
`?hh text0 { text1 }
```

indicates the following actions should be performed when that tag is encountered as part of the HTML code output stream:

- If port D output bit *hh* is low, output the text up to the next {, then discard the text up to the next }.
- If port D output bit *hh* is high, discard the text up to the next {, then output the text up to the next }.

In other words, if bit *hh* is low, output the first *text0* string, else output the second *text1* string. In the example of Listing 1, the text “CHECKED” is output on either the first or second “radio button” HTML code line, depending upon the state of PicoWeb output bit PD4, the output pin which drives the user-controlled LED.

The HTML form shown in Listing 1:

```
<FORM name=mfrm method=GET action="/x">
<input type=radio NAME=4 VALUE=0 `?04 CHECKED{}>on<br>
<input type=radio NAME=4 VALUE=1 `?04{CHECKED}>off<br>
<input type=submit VALUE="Set LED">
</FORM>
```

allows the user of the “WebTemp” PicoWeb Web page to change the state of the user-controlled LED by clicking on the Web page button labeled “Set LED”. When this button is clicked, the Web browser will *submit* an HTTP *GET request* to the PicoWeb server with the following URL:

```
http://x.y.z.w/x?4=s
```

where *x.y.z.w* is the IP address of the PicoWeb server and *s* is the desired state of the LED, either 0 or 1. (The “radio buttons” in the form, depending upon which one is “checked”, cause a *value* named “4” to be sent to the PicoWeb server.) Firmware in the PicoWeb server recognizes HTTP *GET requests* of the form “*x?p=s*”, where *p* is a digit in the range of 0-7, and will set the state of output pin *DBp* to *s*.

Also note that any text strings “*\$\$VERSION\$\$*” will be replaced with the current PicoWeb firmware *version string* when HTML files are processed for downloading into the PicoWeb server’s “file system”.

**Building the Project** - The WebLED project can be built using the PicoWeb development system as in the following examples. The examples that follow assume that commands are being entered into an MS-DOS Prompt Window under Windows 95/98. Note that the environment variable *PWDEV* needs to be set to the full path name of the PicoWeb development system “base directory”, and the PicoWeb development system’s “bin” subdirectory needs to be in the *PATH*. Assuming that the PicoWeb development system “base directory” is *C:\PWDEV*, then the following commands will set up the environment for PicoWeb development:

```
C:>set PWDEV=C:\PWDEV
C:>set PATH=%PATH%;%PWDEV%\bin
```

Before beginning building the WebLED project two text files may need to be edited:

`ip` – must contain the IP address assigned to the PicoWeb server (e.g., 10 . 1 . 2 . 3)  
`ether` – must contain the Ethernet address assigned to the PicoWeb server (e.g., 0 . 1 . 2 . 3 . 4 . 5)

After assigning IP and Ethernet address, the WebLED project can be compiled and readied for download into the PicoWeb server. This can be accomplished with the following command:

```
C:>pwbuild webled
```

This will cause the generation of a number of files, including an assembly language listing file, `webled.lst`, as well as four data files needed to download the firmware and Web pages into the PicoWeb server hardware:

`webled.rom` – Atmel microcontroller program memory binary data (a.k.a., ROM file)  
`webled.ep` – Atmel microcontroller on-chip EEPROM program binary data  
`webled.e1` – PicoWeb CGI pcode binary data, to be loaded into the external serial EEPROM  
`webled.dat` – PicoWeb “Web file system” data (i.e., HTML code and images)

**Downloading the Project** - The first two files can be loaded into the PicoWeb server hardware using a PC parallel port by attaching the PicoWeb programming cable to the connector on the PicoWeb server and to the PC’s parallel port, then entering the following command:

```
C:>pwavrld webled
```

This will erase the program and EEPROM memory in the PicoWeb server’s Atmel microcontroller, then download new firmware and data into the chip.

At this point, if the PicoWeb server is plugged into the network, it should now be active on the Ethernet. In fact, one should be able to “ping” the PicoWeb server from the development system PC using the PicoWeb server’s assigned IP address.

At this point, the Web pages and CGI pcode routines store in the files `webled.dat` and `webled.e1` need to be loaded into the PicoWeb server. This can be done with the following command:

```
C:>pwnetld webled
```

which will download the data in these files over the network using the PC’s network card. Assuming all goes well, the PicoWeb server can now be accessed over the network as a Web server. One should be able to retrieve the Web page shown in Figure 1 by using the following URL:

```
http://x.y.z.w/
```

where `x.y.z.w` is the IP address previously assigned to the PicoWeb server.