Application Note

Thermostat Demo Using eZ80Acclaim!® MCU

AN020603-1008

Abstract

The Thermostat Demo describes how easy it is to develop Internet-enabled process control and monitor applications by embedding Java elements on Zilog's eZ80Acclaim!® microcontroller unit (MCU). A variety of languages are used in the design and specification of hardware and software for embedded electronic systems. The integration of hardware and software components for this Demo is implemented using different languages. The Thermostat drivers described in this application note are written in the C language. A Graphical User Interface (GUI) for the embedded system is developed using Java applets. The Thermostat Demo is thus a Java-enabled process control application supported by Zilog TCP/IP (ZTP) stack that functions on the Internet environment.

With this Java-based demo application, you can monitor and manipulate the Thermostat-based control system in real time and can display dynamic temperature values.

Notes: 1. *The source code file associated with this application note,* AN0206-SC01*.zip, is available for download on* www.zilog.com.

2. *The source code files in this document are intended for use with ZTP software suite v2.0.0 and the Zilog Developer Studio II—IDE for eZ80Acclaim! v4.10.0 (ZDS II v4.10.0).*

Zilog Product Overview

This section provides brief overview of the Zilog products used in this application note. It includes the award-winning eZ80Acclaim! MCU and the full-featured Zilog TCP/IP (ZTP) software suite.

eZ80 Acclaim! MCU Family Overview

The eZ80Acclaim! family of microcontrollers include Flash and non-Flash products. The Flash-based eZ80Acclaim! MCUs, device numbers eZ80F91, eZ80F92, and eZ80F93 are an exceptional value for designing high-performance, embedded, applications. It is possible to have the performance necessary to execute complex applications supporting networking functions quickly and efficiently with speeds up to 50 MHz and an on-chip Ethernet MAC (eZ80F91 only). Combining on-chip Flash and SRAM, eZ80Acclaim! devices provide the memory required to implement communication protocol stacks and achieve flexibility when performing in-system updates of application firmware.

Zilog also offers two eZ80Acclaim! devices without Flash memory (eZ80L92 and eZ80190 microprocessors).

ZTP Software Suite Overview

ZTP Software Suite integrates a rich-set of networking services with an efficient real-time operating system (RTOS). The operating system is a compact preemptive multitasking, multi-threaded kernel, with inter-process communications (IPC) support, and soft real-time attributes. Table 1 lists the standard network protocols implemented as part of the embedded TCP/IP protocol stack in ZTP.

Many TCP/IP application protocols are designed using the client-server model. The final stack size is link-time configurable and determined by the protocols included in the build.

Table 1. Standard Network Protocols in ZTP

HTTP	TFTP	SMTP	Telnet	IP	PPP
DHCP	DNS	TIMEP	SNMP	TCP	UDP
ICMP	IGMP	ARP	RARP	FTP	PPPoE
SNTP	SSL*				

* SSL is available only with SSL package.

Figure 1 displays the flow of data from the embedded environment to the Java-based GUI.

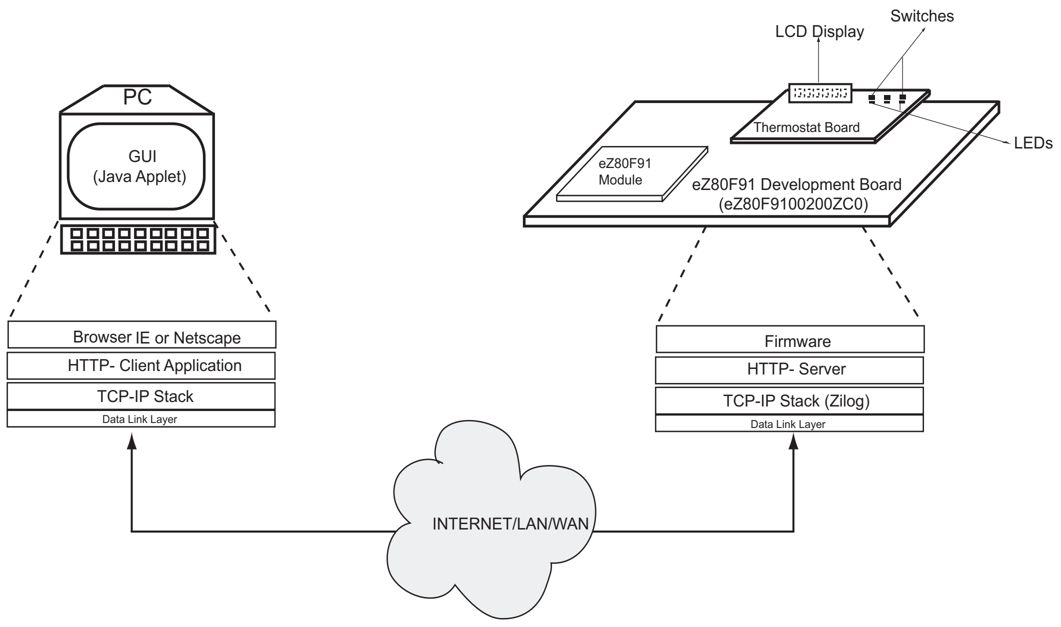


Figure 1. Overview of Thermostat Demo using eZ80Acclaim!®

Discussion

The Thermostat Demo displays how to develop an Internet-based application to control or monitor processes. This is achieved by combining the advantage offered by Java's robustness with a real-time embedded system that talks to physical layer driver.

The architecture and design of the Thermostat Demo incorporates a GUI using Java applets to connect to the embedded environment. The user interface is a Java applet. The real-time and device-specific processes are written in C. The Java GUI obtains the data from the low-level embedded software code (C) and presents the data in a client browser. Figure 1 on page 2 displays the flow of data from the embedded environment to the Java-based GUI.

The Thermostat Demo described in this application note isolates the real-time control sections of an embedded application from its Java-based sections. Thus allowing information to flow freely between the two sections.

An embedded system that uses a Java applet for its GUI requires system software that supports TCP/IP. It is with an HTTP web server running over it that serves HTML pages referring to the Java applet, and the code for the Java applet. The embedded system does not require a Java Virtual Machine. You can access the embedded system via any Java-enabled web browser.

A request to read an HTML page loads a GUI applet into the browser and starts executing it. The applet opens a socket and connects to the main application in the embedded system. The main application, written in C, opens a socket and waits for a connection by the applet. When a connection is made, messages can be sent back and forth between the applet and the main application. This is in response to a request to see data or change settings.

The Java Virtual Machine that executes the GUI applet actually runs on the web browser and not on the embedded system.

Advantages of Using Java

The advantages of using a Java applet are briefly described below.

- The applet is stored as a file in the web server that does not require additional memory from an embedded device to operate; the restricted memory on the embedded system is neatly sidestepped.
- An embedded system vendor can be assured that you have access to a web browser, in any computer platform you choose to access the embedded system from Java applets are platform independent and provide cross-platform GUIs.
- The Java GUI technique works well on a slow network connection, such as a serial line; at a time many clients are served efficiently.
- A Java-based GUI is a much better solution than customized client software, because there is no need to ship any client side media or client side installation instructions with the product. There is no additional cost, because there is no need for OS upgrades or technical support for the client side software. Only one version of the Java-based GUI software is required, and this version is stored in ROM/Flash in the embedded system.

An Embedded HTTP Web server

HTTP web servers use a standard synchronous request or response design running over the TCP/IP, identical to classical client or server architecture. When a client makes a request to an HTTP server, it sends an HTTP request message. The HTTP request message includes the client request, and information about the client's capabilities. The HTTP response is similar to the request, except that it is composed of two parts—the response header and the response body. The response body represents the result of the initial request. A single blank line in this HTTP response file separates the response header from the response body.

The requirements for a web server designed to run on a workstation differs from that designed to run in an embedded system. Features such as the presence of log files, and larger memory footprint in non-embedded web servers, are a definite hindrance when it comes to implementing an embedded web server. Embedded web servers lay emphasis on reducing memory footprint while increasing efficiency, reliability, and providing mechanisms, to generate dynamic data.

Embedded systems do not typically serve a multitude of static web pages. Internet appliances and embedded systems, in general, require web servers that enhance their existing functionality without impinging on vital device resources or requiring a redesign. Because many of these systems are cost-constrained, memory, and CPU resources are usually at a premium. It is vital that embedded web servers offer minimal memory requirements and are efficient.

The requirements for an embedded web server include:

- Memory Usage
- Dynamic Page Refreshing/Update
- Web Page Storage in Flash or ROM

Memory Usage

One of the most important requirements for an embedded web server is small memory footprint. The web server use very little memory (code, stack, and heap), and it must not fragment memory. Many embedded devices employ simple memory allocations that cannot combine fragmented memory effectively. Because web servers must often respond to requests to serve pages, simple memory allocations can cause problems. When the memory used to serve a page is freed, it can be useless, as it cannot be merged with adjacent memory blocks on the heap. To solve this problem, embedded web servers should use only statically allocated or pre-allocated memory blocks.

Dynamic Page Refreshing/Update

An embedded device features only a small number of pages in memory, and often refreshes the page contents on the fly. The web pages display ever-changing information about device status, values read by sensors, and any other data available to the device.

Web Page Storage in Flash or ROM

Many embedded systems do not feature disk drives, yet they must be accessed and controlled via the web. In such cases, a method of storing web pages in ROM is required. Embedded web servers should be able to access HTML, Java applets, image files, and any other web contents stored in Flash Memory or ROM.

Reading and Writing to the Embedded Web server

To communicate over a network using Java programs, the socket or uniform resource locator (URL) classes provided in the java.net package are used. The underlying TCP and UDP layers are not a concern. Socket is one end-point of a two-way communication link between two programs running on the network. URL is a pointer to a resource on the World Wide Web.

The java.net package provides two socket-related classes:

1. Class Socket–That implements the client side of a network connection.
2. Class ServerSocket–That implements the server side of a network connection

Class Socket implements the client side of a two-way connection between the Java program and another program on the network, while Class

ServerSocket provides a system-independent implementation of the server side of a client/server socket connection. The Thermostat Demo implements only the Socket class because the server program is implemented in C, and not in Java.

Java programs that interact over the Internet can also use URLs to find Internet resources. Java programs use Class URL in the java.net package to represent a URL address. Java applets use a URL to reference and connect to these network resources. If the server supports a protocol that a URL recognizes (for example, http), the URL can be used to create a URL Connection to the server (which normally connects via the Port 80 socket) to manage protocol-specific communication.

Sockets can be used regardless of whether the server supports a protocol that recognizes a URL or not, but while using a socket to contact an HTTP server, some of the details managed by the URL must be entered manually. To avoid using established protocols, communication according to user specifications can be effectively managed via sockets.

For more information on sockets and URLs, see References on page 19.

Thermostat Implementation Using eZ80Acclaim!®

To understand the Thermostat implementation, it is necessary to know how the temperature is controlled by the eZ80Acclaim! MCU.

Temperature Control

This section discusses how the eZ80Acclaim! processor controls the temperature. The Java applet features buttons that allow you to send commands to the processor to control the temperature around the temperature sensor. It is for setting the Thermostat control parameters, such as upper and lower temperature set points. Figure 3 on page 8 displays a screenshot of the GUI implemented using a Java applet.

The upper and lower set point values are passed to the embedded HTTP server using the Java socket connections that invoke the CGI script. The processor obtains these values via the firmware interface that use the CGI script. The processor continuously (every two seconds) reads the temperature of the sensor and switches on/off the bulb or the fan to maintain the temperature within these set limits. The new temperature values are sent via the CGI-Firmware interface to the HTTP web server. Here the temperature values in the Java applet are updated and finally displayed on the screen. The processor also updates the temperature values on the liquid crystal display (LCD) unit every two seconds.

Hardware Architecture

Figure 2 is a block diagram of the hardware architecture featuring the eZ80® development platform and the Thermostat Board.

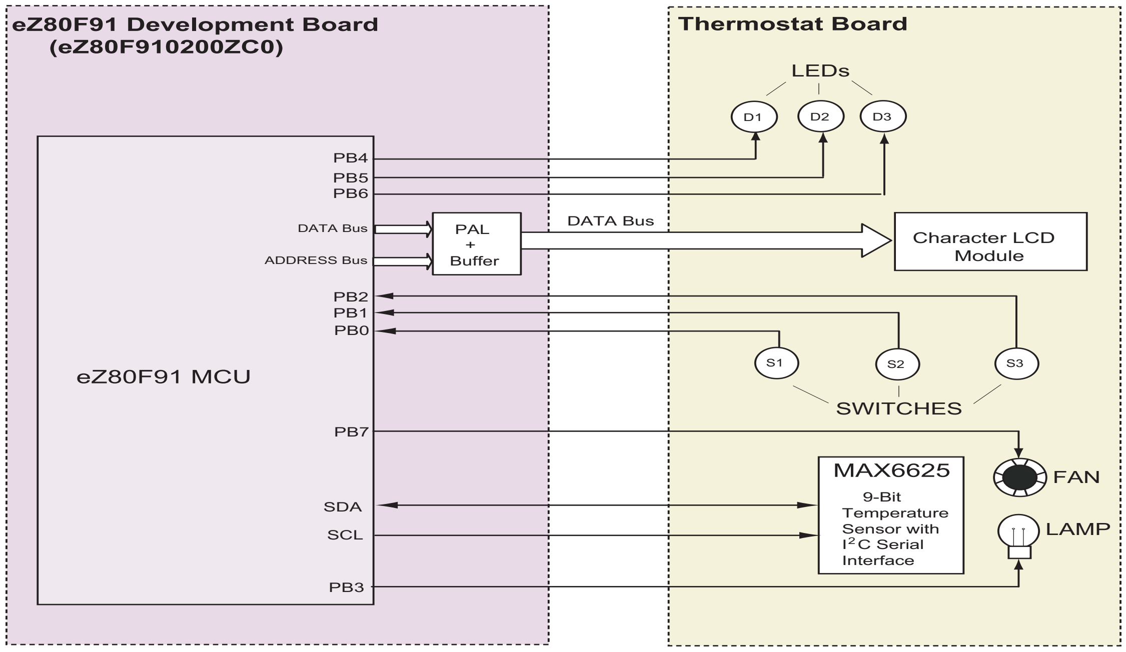


Figure 2. Hardware Block Diagram for Thermostat Demo

The pins PB4, PB5, and PB6 on the eZ80F91 MCU are connected to the LEDs on the Thermostat Board. The LCD Module is used as a memory map device, with 0x800002 as the memory address. The programmable array logic (PAL) unit is used for address decoding. For details of the LCD connection, see Figure 2. The LCD module displays the changing temperature of the temperature sensor dynamically.

The pins PB0, PB1, and PB2 are connected to three press button switches, S1, S2, and S3 respectively. These switches can be used to force heating or cooling or prevent heating or cooling of the temperature sensor. Pin PB7 connects to the fan that cools the temperature sensor when turned on, while pin PB3 connects to a lamp that heats up the temperature sensor when turned on.

MAX6625 is a 9-bit Temperature Sensor with an I^2C Serial interface that is connected to the SDA and SCL lines on the eZ80F91 MCU.

Software Implementation

The software implementation for the Java-based Thermostat Demo is divided into the following sections:

Development of Java Applets—This section explains the implementation of the GUI in the form of several Java applets to create a socket connection, display temperature reading, allow setting of the upper and lower temperature values, indicate the status of heating/cooling as controlled by the hardware switches, and send inputs to control the LEDs.

Development of the Embedded Firmware—

This section explains the two-sided firmware interface. One side interfaces with the CGI script inputs from the HTTP web server, while the other side interfaces with the actual hardware devices like the temperature sensor and the LCD panel.

Adding and Integrating Thermostat Demo Files to ZTP—The software implementation for the Java-based Thermostat Demo involves the use of the ZTP stack. The eZ80F91 Development Kit contains the ZTP stack software that supports socket connections. This section contains the details for adding and integrating the Thermostat Demo application with the ZTP stack. For more details on the ZTP stack refer to www.zilog.com and contact the Zilog help desk.

Development of Java Applets

Java applets are developed using a GUI builder (any text-based editor like Notepad can be used). Sun's Java Development Kit (JDK) is used for compiling the applet source files.

The Java applet, TstatHttpClient.class creates the socket connection to communicate to the host computer from where the applets are loaded. When any applet requires information from the main application, it sends a request message via this socket. The main application responds by sending a response message through the same socket.

The rest of the Java applets provide control and monitoring capabilities to the Thermostat Board on the eZ80® development platform. The Java applets provide the following functionality:

- The facility to manipulate the upper and lower set points for the temperature sensor.
- The facility to display the updated temperature values on a client web browser in graphical and numerical formats.
- The facility to display the status of the hardware switches, which are used to force heating or cooling or turn off the heating and cooling, on the temperature sensor.

Development of the Embedded Firmware

The embedded firmware interfaces at two levels:

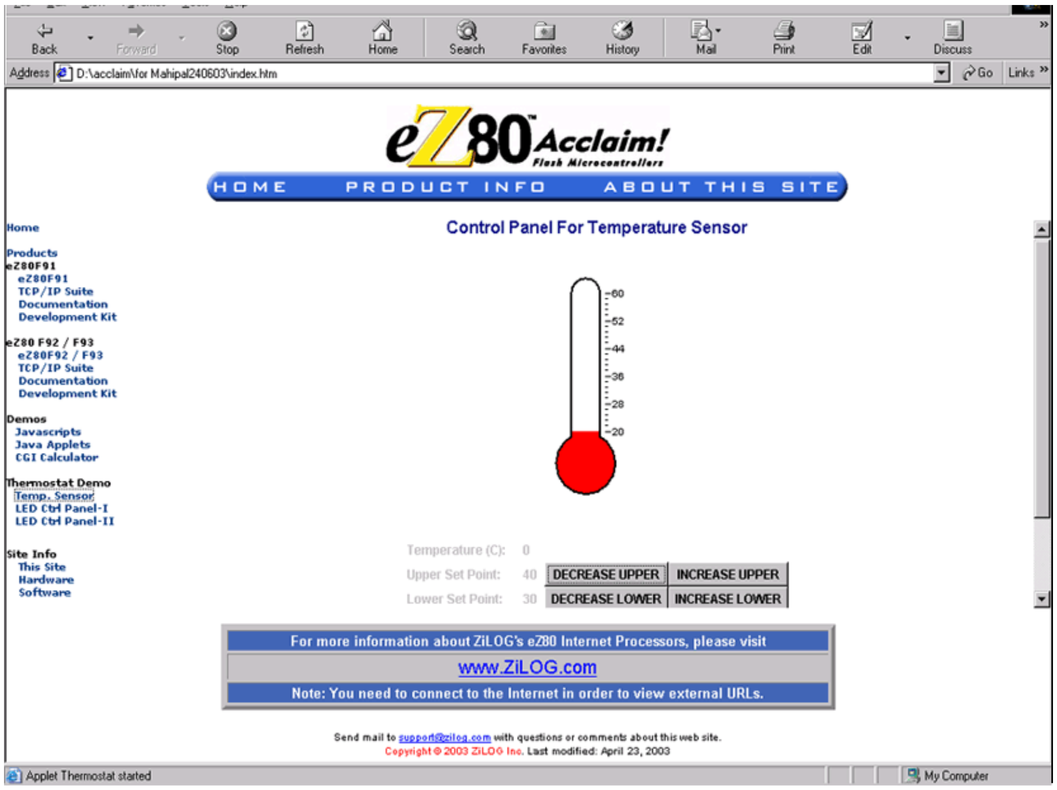
1. At one level it interfaces between the HTTP web server on ZTP and the Java applets.
2. At another level the embedded firmware interfaces with the actual hardware devices on the Thermostat Board and the LCD panel.

The following sub-sections explain the details of the embedded firmware interfaces.

Firmware Interface to Java Applets

The firmware interface establishes communication links between the Java GUI and the firmware. This firmware interface is written in C and developed using the ZDS II.

Figure 3 is a screen shot of the web page using the Thermostat applet. It displays the Thermostat.class applet, receives the temperature data, and displays it via the applet, on the web browser. The applet invokes the web server's CGI scripts contained in the tstat_control_cgi.c to write to and read from the HTTP web server.



Control Panel For Temperature Sensor

Temperature (C): 0
Upper Set Point: 40
Lower Set Point: 30

For more information about ZiLOG's eZ80 Internet Processors, please visit www.ZiLOG.com

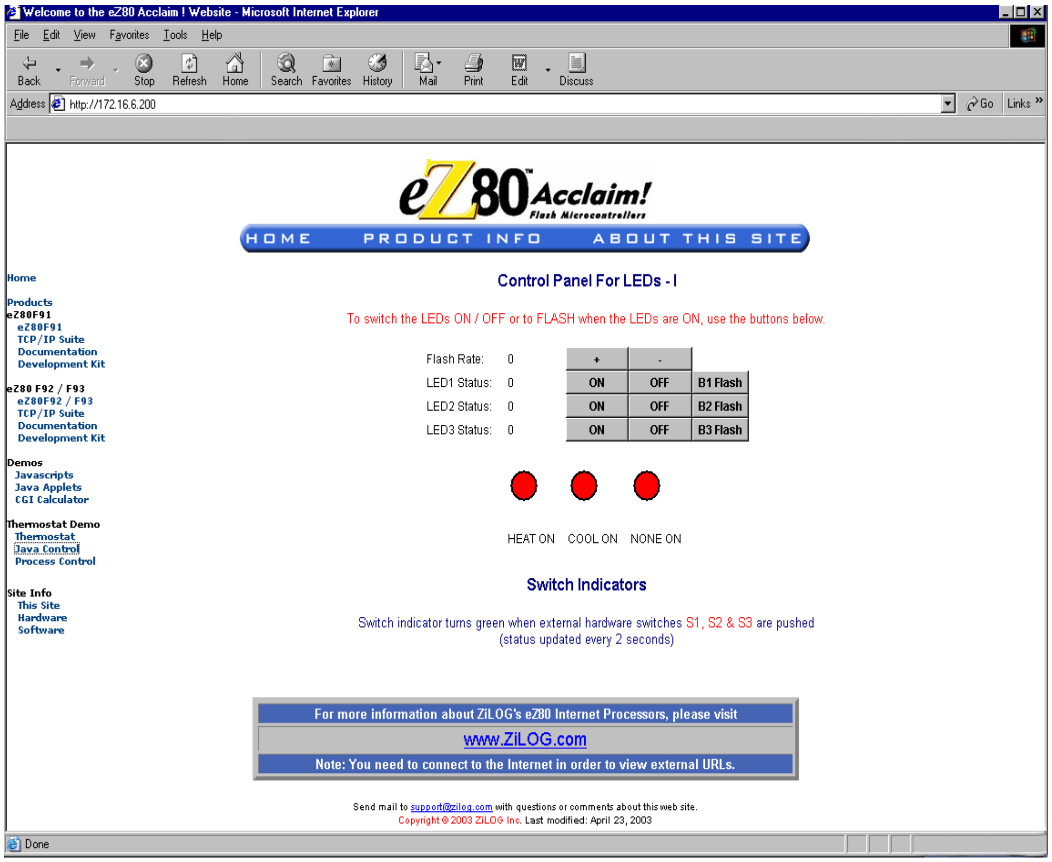
Note: You need to connect to the Internet in order to view external URLs.

Send mail to support@zilog.com with questions or comments about this web site.
Copyright © 2003 ZiLOG Inc. Last modified: April 23, 2003

Figure 3. Control Panel for Temperature Sensor

Figure 4 is a screen shot of the web page using the Button applet. It displays the Button.class applet, demonstrates how the LEDs are controlled remotely, and how the hardware-controlled Thermostat heating and cooling is displayed remotely.

The Button applet uses the CGI script file, java_control_cgi.c to control the LEDs and check the hardware switch inputs.


Welcome to the eZ80 Acclaim ! Website - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://172.16.6.200
Go Links
eZ80 Acclaim! Flash Microcontrollers
HOME PRODUCT INFO ABOUT THIS SITE
Home
Products
eZ80F91
eZ80F91
TCP/IP Suite
Documentation
Development Kit
eZ80 F92 / F93
eZ80F92 / F93
TCP/IP Suite
Documentation
Development Kit
Demos
Javascripts
Java Applets
CGI Calculator
Thermostat Demo
Thermostat
Java Control
Process Control
Site Info
This Site
Hardware
Software
Done

Control Panel For LEDs - I

To switch the LEDs ON / OFF or to FLASH when the LEDs are ON, use the buttons below.

Flash Rate:	0	+	-	
LED1 Status:	0	ON	OFF	B1 Flash
LED2 Status:	0	ON	OFF	B2 Flash
LED3 Status:	0	ON	OFF	B3 Flash

HEAT ON COOL ON NONE ON

Switch Indicators

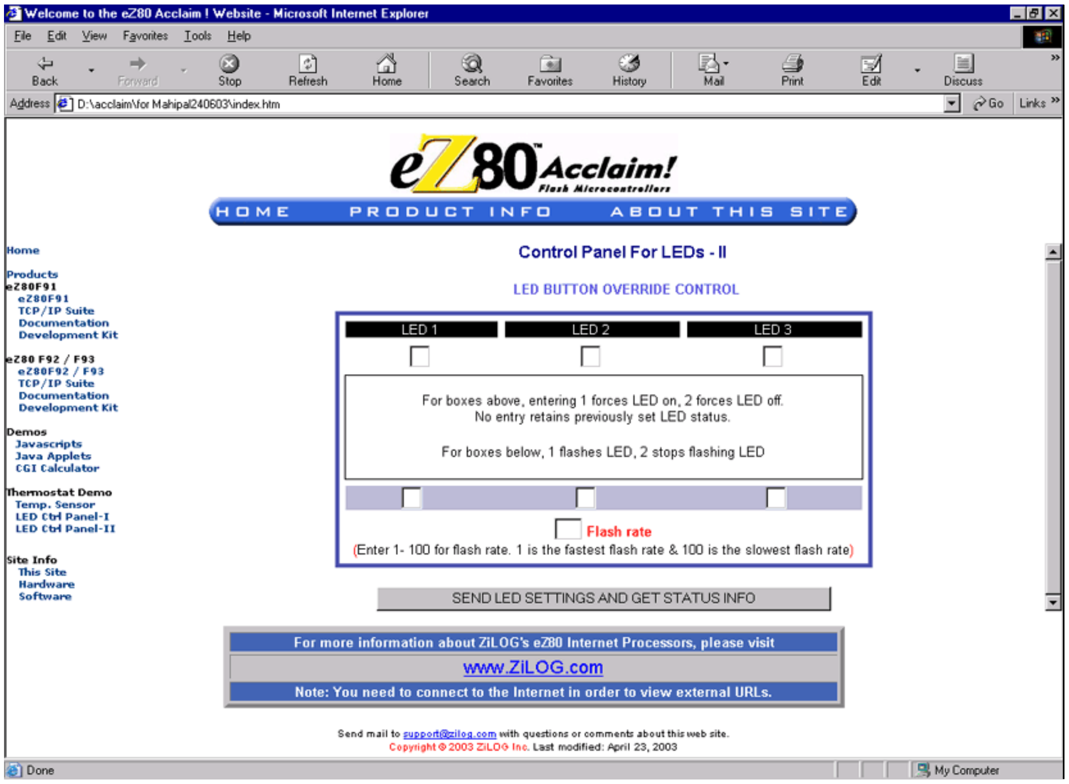
Switch indicator turns green when external hardware switches S1, S2 & S3 are pushed
(status updated every 2 seconds)

For more information about ZiLOG's eZ80 Internet Processors, please visit
www.ZiLOG.com
Note: You need to connect to the Internet in order to view external URLs.

Send mail to support@zilog.com with questions or comments about this web site.
Copyright © 2003 ZiLOG Inc. Last modified: April 23, 2003

Figure 4. Control Panel for LEDs-I

Figure 5 is a screen shot of the web page generated by an HTML form. It displays the LEDs controlled by inputs into this HTML form. The HTML page, **Control panel for LEDs-II** demonstrates ON/OFF and Flash rate control using HTML forms. The HTML form handler uses the CGI script file switches_cgi.c.


Welcome to the eZ80 Acclaim ! Website - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address D:\acclaim\for Mahipal240603\index.htm
Go Links
eZ80 Acclaim! Flash Microcontrollers
HOME PRODUCT INFO ABOUT THIS SITE
Home
Products
eZ80F91
eZ80F91
TCP/IP Suite
Documentation
Development Kit
eZ80 F92 / F93
eZ80F92 / F93
TCP/IP Suite
Documentation
Development Kit
Demos
Javascripts
Java Applets
CGI Calculator
Thermostat Demo
Temp. Sensor
LED Ctrl Panel-I
LED Ctrl Panel-II
Site Info
This Site
Hardware
Software
LED 1
LED 2
LED 3
Flash rate
SEND LED SETTINGS AND GET STATUS INFO
Done
My Computer

Control Panel For LEDs - II

LED BUTTON OVERRIDE CONTROL

For boxes above, entering 1 forces LED on, 2 forces LED off.
No entry retains previously set LED status.

For boxes below, 1 flashes LED, 2 stops flashing LED

(Enter 1- 100 for flash rate. 1 is the fastest flash rate & 100 is the slowest flash rate)

For more information about ZiLOG's eZ80 Internet Processors, please visit
www.ZiLOG.com
Note: You need to connect to the Internet in order to view external URLs.

Send mail to support@zilog.com with questions or comments about this web site.
Copyright © 2003 ZiLOG Inc. Last modified: April 23, 2003

Figure 5. Control Panel for LEDs-II

Embedded web server connections are established using either the Socket Class or the URL Class available in the java.net package. The URL code segment that can be used instead of a direct socket connection to a server is explained here. To write to a server, a Port 80 socket connection to the server is created. The code to obtain this socket connection is called via the getupdate ( ) method, and the code to write to the socket is called via a sendRequest ( ) method.

To read dynamic data, a URL to the CGI dynamic page is built on the server from where the applets are delivered. After opening the URL connection, the data is received in this connection. When all of the data is sent through an output stream (by calling the getupdate ( )method), an input stream (in Stream) for the URL connection reads the server's response.

The CGI interface programs make use of the HTTP functions http_init( ), http_get( ), http_post( ), and http_request( ).These programs are common functions used to build the embedded HTTP web server. The CGI script file, input_cgi.c, passes the maximum and minimum set point values from the Java applet to the main.c program. The functions in the main.c file, calls the appropriate functions to read the temperature values from the temperature sensor on the Thermostat Board.

It maintains the temperature between the upper and lower set points, (by turning on the bulb or the fan) by dynamically upgrading the limits you set using the buttons on the GUI.

The constantly changing temperature values from the temperature sensor are read by the Java applet by invoking the CGI script, and displayed in a graphical and numerical format on the web browser. Reading and writing to the server occur every two seconds. A thread running in the applet controls this process.

Firmware Interface to Hardware

The firmware interface to the hardware performs the following tasks:

- Initializes the eZ80Acclaim!® I/O ports to configure them for reading and writing to the devices on the Thermostat Board and the LCD panel.
- The firmware interface calls the function http_init ( ) with appropriate parameters to build the web server. The web server creates several threads so that multiple web servers run on multiple ports. The structure, web pages, defines the kinds of pages that are embedded in the website. All necessary static and dynamic web pages that are built in this structure are defined. The index.html and tstat_control.html are created as dynamic web pages and the Thermostat.class is created as a static web page.
- The firmware interface initializes the I^2C temperature sensor.
- Reads the temperature from the MAX6625 Temperature Sensor using the readtemp ( ) function.
- The Firmware interface is used to exchange data between the temperature sensor and a LCD panel, to display temperature variation dynamically. The LCD panel is plugged to the port A of the eZ80F91 MCU. The LCD program initially displays the IP address of the server. When you request a URL via the browser, the program displays the set upper/ lower temperature values, and the current temperature on the LCD panel. The temperature reading is updated every two seconds.

Adding and Integrating Thermostat Demo Files to ZTP

The Thermostat Demo described in this application note requires the eZ80Acclaim!® development board, ZTP stack, and the Thermostat Board. For the Thermostat Demo execution, the files specific to the Demo must be added and integrated to the ZTP stack before it is downloaded onto the eZ80Acclaim! development board. This section contains the details of adding the Thermostat Demo files to the ZTP stack.

The Thermostat Demo files that must be added to the ZTP project are located in AN0206-SC01.zip file available for download on www.zilog.com. The Demo files are of the following types:

- C (*.c) files
- Header (*.h) files
- HTML (*.htm) files
- Java (*.class) files

The ZTP stack is available on www.zilog.com and can be downloaded to a PC, with a user registration key. ZTP can be installed in any location; its default location is C:\Program Files\Zilog.

Note: *For ZDS II and ZTP version used in this application, see* Requirements *on page 17.*

Follow the steps below to add and integrate the Demo files to the ZTP stack:

1. Download ZTP. Browse to the location where ZTP is downloaded, and open the ..\ZTP_2.0.0\ZTP\SamplePrograms\ZTPDemo folder.
2. Install eZ80F91—ZTP Projects Library available under Application Sample Libraries from www.zilog.com. Copy all the *.htm (or *.html) and *.class files under Thermostat Demo application library located in the ...\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\Thermostat Demo Application\include\TD_Website.Acclaim, folder to the ...\ZTP_2.0.0\ZTP\SamplePrograms\Website.Acclaim folder.
3. Copy all the *.c, *.s files located in ...\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\Thermostat Demo Application\source and *.h files located in ...\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\Thermostat Demo Application\include folder to the ...\ZTP_2.0.0\ZTP\SamplePrograms\ZTPDemo folder.
4. Launch ZDS II for eZ80Acclaim! v4.10.0 and open the website.zdsproj project located in the path: ...\ZTP_2.0.0\ZTP\SamplePrograms\website.Acclaim.
5. Add all the *.htm (or *.html) and *.class files located in the ..\website.Acclaim folder to the website.zdsproj project. To do so, click **Project** and then click **Add Files**. The *.htm files to be added are listed below:

   ```
   control_page.htm
   jcontrol_page.htm
   ThermostatDemo.html
   thermostatf.htm
   ```

```
tstat_control_page.htm
```

The *.class files to be added are listed below:

```
ButtonApplet.class
CustomParser.class
LEDBulb.class
messagerA.class
MiniHttpClient.class
ParamParser.class
Thermometer.class
Thermostat.class
TstatHttpClient.class
```

6. Open the website file from within ZDS II, and add the following prototype declarations to it:

```
// Thermostat pages
extern struct staticpage control_page_htm;
extern struct staticpage jcontrol_page_htm;
extern struct staticpage thermostatf_htm;
extern struct staticpage thermostat_htm;
extern struct staticpage ThermostatDemo_html;
extern struct staticpage tstat_control_page_htm;
extern int input_cgi (struct http_request *request);
extern int java_control_cgi (struct http_request *request);
extern int switches_cgi (struct http_request *request);
extern int Thermostat_cgi (struct http_request *request);
// Java Applets
extern struct staticpage Thermometer_class;
extern struct staticpage Thermostat_class;
extern struct staticpage LEDBulb_class;
extern struct staticpage ButtonApplet_class;
extern struct staticpage CustomParser_class;
extern struct staticpage messagerA_class;
extern struct staticpage MiniHttpClient_class;
extern struct staticpage ParamParser_class;
extern struct staticpage TstatHttpClient_class;
```

7. The website file contains an array, web page website[], that provides information about the HTML pages. Replace the last line of the array, {0, NULL, NULL, NULL} with the following lines of code:

```
{HTTP_PAGE_STATIC,"/Thermometer.class", "application/octect-stream",
&Thermometer_class},
{HTTP_PAGE_STATIC,"/Thermostat.class", "application/octect-stream",
&Thermostat_class},
{HTTP_PAGE_STATIC,"/LEDBulb.class", "application/octect-stream",
&LEDBulb_class},
{HTTP_PAGE_STATIC,"/ButtonApplet.class", "application/octect-stream",
&ButtonApplet_class},
{HTTP_PAGE_STATIC,"/CustomParser.class", "application/octect-stream",
&CustomParser_class },
{HTTP_PAGE_STATIC,"/messagerA.class", "application/octect-stream",
&messagerA_class},
{HTTP_PAGE_STATIC,"/MiniHttpClient.class", "application/octect-stream",
&MiniHttpClient_class},
{HTTP_PAGE_STATIC,"/ParamParser.class", "application/octect-stream",
&ParamParser_class},
{HTTP_PAGE_STATIC,"/TstatHttpClient.class", "application/ octect-
stream", &TstatHttpClient_class},
{HTTP_PAGE_STATIC,"/control_page.htm", "text/html", &control_page_htm},
{HTTP_PAGE_DYNAMIC,"/cgi-bin/switches", "text/html", (struct
staticpage*) &switches_cgi},
{HTTP_PAGE_DYNAMIC,"/cgi-bin/java_control","text/html",(struct
staticpage*)&java_control_cgi},
{HTTP_PAGE_DYNAMIC,"/Thermostat.html", "text/html", (structstatic
page*)&Thermostat_cgi},
{HTTP_PAGE_STATIC,"/jcontrol_page.htm", "text/html",
&jcontrol_page_htm},
{HTTP_PAGE_STATIC,"/thermostatf.htm", "text/html", &thermostatf_htm},
{HTTP_PAGE_STATIC,"/tstat_control_page.htm", "text/html",
&tstat_control_page_htm},
{HTTP_PAGE_DYNAMIC, "/Data.html", "text/html", (struct staticpage*)
&input_cgi},
{0, NULL, NULL, NULL}
```

8. From within ZDS II v4.10.0, open the left.htm file located in the \Web Files folder. Search for the Demos<br> statement and add the following piece of HTML code above the Demos<br> statement to create a link from the default *eZ80Acclaim!®* web page to the *Thermostat Demo* web page.

```
Thermostat Demo<br> &nbsp; &nbsp;
<a href="tstat_control_page.htm">Temp.Sensor</a><br> &nbsp;&nbsp;
<a href="jcontrol_page.htm" target="main">LED Ctrl Panel-I</a><br>
&nbsp; &nbsp;
<a href="control_page.htm" target="main">LED Ctrl Panel-II</a><br>
&nbsp; &nbsp; &nbsp;
<br>
```

9. Build the website.zdsproj project to obtain the new library file Acclaim_website.lib. Copy this Acclaim_website.lib to the
   . . .\ZTP_2.0.0\ZTP\Lib folder where it will replace the existing one.
10. Close the website.zdsproj project.
11. From within ZDS II, open the ZTPDemo_F91.zdsproj file available in the path
    . . .\ZTP_2.0.0\ZTP\SamplePrograms\ZTPDemo.
12. Add all the *.c files located in the . . .\ZTP_2.0.0\ZTP\SamplePrograms\ZTPDemo folder to the ZTPDemo_F91.zdsproj project. To do so, click **Project** and then click **Add Files**.

    The *.c files to be added are listed below:

```
initialization.c
input_cgi.c
java_control_cgi.c
LCD_API_port.c
switches_cgi.c
```

```
temp_read.c
Thermostat_cgi.c
tstat_control_cgi.c
```

The *.s files to be added are listed below:

```
Timer1ISRProlog.s
Timer2ISRProlog.s
```

13. For this application, dynamic host configuration protocol (DHCP) is disabled; therefore, ensure that in the ZTPConfig.c file

```
UINT8 b_use_dhcp = FALSE
```

14. Look for the ifTbl structure definition in the ZTPConfig.c file.

```
struct commonServers csTbl=
{
"172.16.6.28",              // Default Timer Server
"",                         // Default rfs server
"",                         // Default File Server Not currently Used
"172.16.6.194"              // Default Name Server
};

struct If ifTbl [MAX_NO_IF] =
{
// interface 0 -> EthernetConfiguration
{
&usrDevBlk [0],             //Control block for this device
ETH,                        // interface type
ETH_MTU,                    // MTU
ETH_100,                    // Speed can be ETH_10 or AUTOSENCE
"172.16.6.198",             // Default IP address
"172.16.6.1",               // Default Gateway
0xffff0000UL                // Default SubnetMask
}
}
```

Note: *The* ifTbl *structure contains network parameters and settings (in the four-octet dotted decimal format) specific to the local area network at Zilog, by default. Modify the above structure definition with appropriate IP addresses within your local area network (For details on modifying the structure definition, refer to the ZTP v2.0.0 documents).*

15. Open the emac_conf.c file and change the default MAC address (provided by ZTP) such that each eZ80® development platform on the LAN contains a unique MAC address. For example,

```
const CHAR f91_mac_addr [ETHPKT_ALEN] = {0x00, 0x90, 0x23, 0x00, 0xDF,
0x91};
```

In the six byte MAC address shown above, the first three bytes must not be modified; the last three bytes can be used to assign a unique MAC address to the eZ80Acclaim!® development platform.

16. Open the main.c file of the ZTPDemo_F91.zdsproj project, and add the following include file.

```
#include "LCD_API.h"
```

```
#include <netif.h>
#include "emulator.h"
```

Add two more lines mentioned below.

```
RZK_THREADHANDLE_t RZKGetCurrentThread ();
RZK_STATUS_t status;
```

17. Add the following function prototypes and global variables to the main.c file.

```
//prototype functions
extern void reg_init_function(void);
extern void tstat_function();
//global declarations
unsigned char flash_mask,jflash_mask;
unsigned char flash_rate,
 flash_speed_entry,jflash_rate,jflash_speed_entry;
Unsigned char LED1_status, LED2_status, LED3_status, jout_hold;
int ambient_temp,upper_setpoint,lower_setpoint,j;
char temp_rising,bypass_counter;
unsigned char io_hold,io_work;
long delay_count=0; int temp;
int temp_low_byte,temp_high_byte,temp_degrees_f, temp_degrees_c;
int i2c_shiftreg,i2c_count,i2c_error, ok;
```

18. Add the following lines of code above the return (OK) statement located at the end of the ZTPAppEntry function in main.c file.

```
reg_init_function ();
LCD_init ();
LCD_prints ("Zilog Acclaim!");// Print a string
LCD_setposition (1, 0);
LCD_prints ("IP");
LCD_setposition (1, 3);
LCD_prints (ifTbl [0].myip);  // Print a string
while(1)
{
tstat_function ();
status = RZKSuspendThread(RZKGetCurrentThread(), 100);
}
```

19. Comment out the following line in the main.c file.

```
Initialize_FileSystem ();
```

Also comment the Intialize_Filesystem ( )function description in the main.c file.

20. In the ZTPInit_Conf.c file, comment the following line:

```
Init_DataPersistence ();
```

21. In kernel.h file comment the following line:

```
#define OK 1
```

22. In ZTPConfig.c file make g_ShellLoginReqd = FALSE
23. Save the files and close the ZTPDemo_F91.zdsproj project.

Demonstration

This section provides the requirements and instructions to set up the Thermostat Demo and run it.

Requirements

The requirements are classified under hardware and software.

Hardware

- eZ80F91 Development Kit (eZ80F910200ZCO)
- Thermostat Board (eZ801900100ZAC) along with a 9 V power supply
- PC with an Internet browser

Software

- Zilog Developer Studio II—IDE for eZ80Acclaim!® v4.10.0 (ZDS II-IDE v4.10.0).
- Zilog TCP/IP Software Suite v2.0.0 (ZTP v2.0.0).
- Project file for the Thermostat Demo for eZ80Acclaim! located at the installed path of Application library under the folder . . .\Applications_Library\eZ80F91_ZTP_Projects_Library 1.0.0\Thermostat Demo Application\.

Setup

This particular setup uses ZPAK II. For demonstration setup with other debug tools, for example, USB Smart Cable, Ethernet Smart Cable etc, refer to their respective user manual.

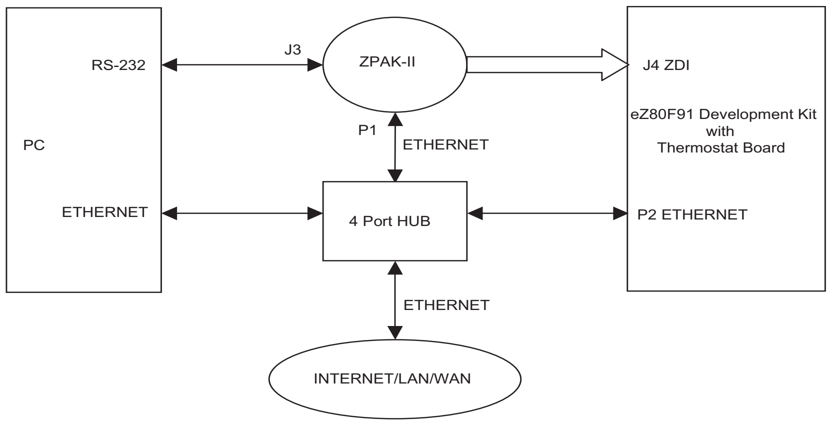


Figure 6. Setup for Thermostat Demo

Settings

This section lists the HyperTerminal settings and Jumper settings for the Thermostat Demo application described in this document.

HyperTerminal Settings

Set HyperTerminal to 57.6 kbps baud, 8-N-2, with no flow control

Jumper Settings

For the eZ80® development platform

- J11, J7, J2 are ON
- J3, J20, J21, J22 are OFF
- For J14, connect 2 and 3
- For J19, MEM_CEN1 is ON, and CS_EX_IN, MEM_CEN2, and MEM_CEN3 are OFF

For the eZ80F91 Module mounted on the eZ80® development platform

JP3 is ON

For the Application Board

For J10, connect the center pin to V_{CC}

Procedure

Follow the procedure below to build and run the Thermostat Demo.

1. Ensure that the required Thermostat Demo files are added and integrated to ZTP before proceeding. For details, see Adding and Integrating Thermostat Demo Files to ZTP on page 12.
2. Make the connections as displayed in Figure 6 on page 17. Follow the jumper settings provided in Jumper Settings on page 17.
3. Connect the 9 V power supply to the eZ80F91 Development Kit and the Thermostat Board separately.
4. Connect the 5 V power supply to ZPAK II and the 7.5 V power supply to the Ethernet hub.
5. Launch the HyperTerminal and follow the settings provided in the HyperTerminal Settings on page 17.
6. From within the HyperTerminal, press *z* repeatedly, and then press the **Reset** button on ZPAK II to view the menu to set the ZPAK II IP address.
7. Enter *H* to display help menu, and follow the menu instructions to obtain the IP address for ZPAK II in order to download the Demo file. This ZPAK II IP address must be entered in the ZDS II.
8. Launch ZDS II for eZ80Acclaim!® and open the Thermostat Demo project file (ZTPDemo_F91.zdsproj) located in the path: . . .\ZTP_2.0.0\ZTP\SamplePrograms\ZTPDemo
9. Open the ZTPConfig.c file. Ensure that the ifTbl structure contains information that is relevant to your network configuration.
10. Build the project and download the resulting file to the eZ80F91 Module on the eZ80 development platform using ZDS II.
11. Run the Thermostat Demo.

Running the Thermostat Demo

1. Launch the Internet Browser on the PC. Enter the IP address specified in the ZTPConfig.c file. The Index.html page is displayed.
2. Click on the **Temp. Sensor** link in the left pane. The **Control Panel for Temperature Sensor** page is displayed. Observe the temperature displayed in graphical and numerical form. Observe the upper and lower limits set for the temperature.
3. Click on the **DECREASE UPPER/ DECREASE LOWER/INCREASE UPPER/ INCREASE LOWER,** buttons to change the upper/lower limits of the temperature. Observe the temperature reading.
4. On the Demo Thermostat Board, hold down switch **S1.** The bulb glows, and heats the temperature sensor. Observe that the temperature reading rises above the set upper limit as long as the S1 switch is held down.
5. Hold down switch **S2**. The fan rotates, and cools the temperature sensor. Notice that the temperature reading falls below the set lower limit as long as the S2 switch is held down.
6. Hold down switch **S3.** Neither the light bulb nor the fan works. Notice that the temperature reading reaches the ambient temperature irrespective of the set upper and lower limits and remains steady as long as the S3 switch is held down.
7. Click on the **LED Ctrl Panel-I** link in the left pane of the Browser window. The **Control Panel for LEDs-I** page is displayed.
8. Switch on the LEDs using the **ON** button. Click on the Flash Rate buttons to specify the

rate for blinking, and click the **Flash** button to activate blinking. On the Board, notice that all the LEDs are on and are blinking at the rate specified.

9. Switch off the LEDs using the **OFF** button. Notice that the LEDs on the Board are switched off.
10. On the Demo Thermostat Board, hold down switch **S1.** The bulb glows. The Switch indicator for **HEAT ON** turns green, indicating that the S1 switch on the Thermostat Board is ON. Repeat this for the remaining switches and observe the effects.
11. Click on the **LED Ctrl Panel-II** link in the left pane of the Browser window. The **Control Panel for LEDs-II** page is displayed.
12. Enter **1** in all the fields, to turn the LEDs ON.
13. In the **Flash** text box, enter **1** to make the LEDs blink. Turning the blinking on is possible only for those LEDs that are ON.
14. In the Flash rate text box, enter a number between **1–100** to set the blinking at a specified rate. Entering 1 sets the blinking to the fastest rate while entering 100 sets it to the slowest rate.
15. Click the **Send LED Settings and Get Status Info** button to effect the changes and obtain an LED status report.

Summary

This application note highlights the eZ80Acclaim!® MCU's capability to perform as efficient embedded web servers by demonstrating an Internet-enabled process control. It monitors application in the form of a Thermostat Demo that embeds Java elements on the eZ80Acclaim! microcontroller.

The Thermostat Demo is a Java-enabled process control application that uses the Internet. With this Java-based demo application, it is possible to monitor and manipulate a control system in real time and display values dynamically.

The advantage of using eZ80Acclaim! with Java is that there is platform independence with a Java GUI as a client application. The real-time processes are controlled and monitored by the eZ80Acclaim! MCU interacting with the hardware devices.

References

The documents associated with eZ80F91 MCU, ZDS II, ZPAK II, and the Thermostat Board available on www.zilog.com are provided below:

- eZ80® CPU User Manual (UM0077)
- eZ80F91 Flash MCU Product Specification (PS0192)
- Thermostat Application Module Product User Guide (PUG0014)
- Zilog Developer Studio II—eZ80Acclaim!® User Manual (UM0144)
- ZPAK II Debug Interface Tool Product User Guide (PUG0015)
- Zilog TCP/IP Stack API Reference Manual (RM0040)
- Zilog TCP/IP Software Suite Programmers Guide (RM0041)
- Zilog TCP/IP Software Suite Quick Start Guide (QS0049)
- Information about sockets and URLs http://java.sun.com/j2se

Appendix A—Flowcharts

This appendix provides the flowcharts for the Thermostat Demo implementation on the eZ80F91 MCU.

Figure 7 displays the flowchart for the main routine.

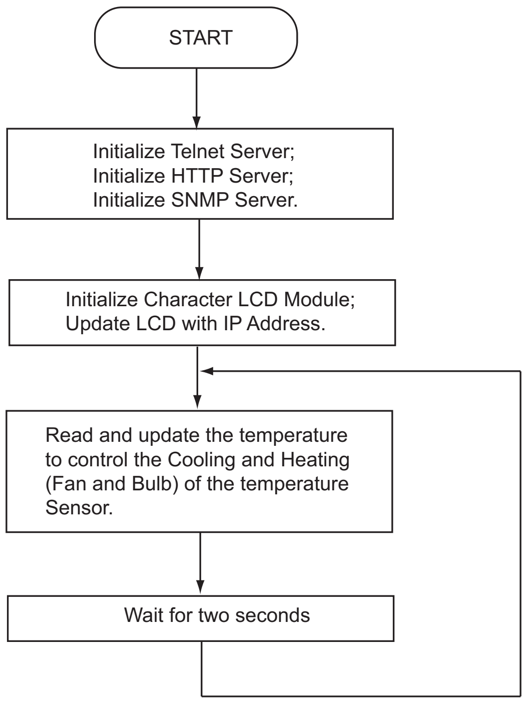

START
Initialize Telnet Server;
Initialize HTTP Server;
Initialize SNMP Server.
Initialize Character LCD Module;
Update LCD with IP Address.
Read and update the temperature to control the Cooling and Heating (Fan and Bulb) of the temperature Sensor.
Wait for two seconds

Figure 7. Flowchart for the Main Routine

Figure 8 displays the flowchart for the Thermostat Applet and LCD update.

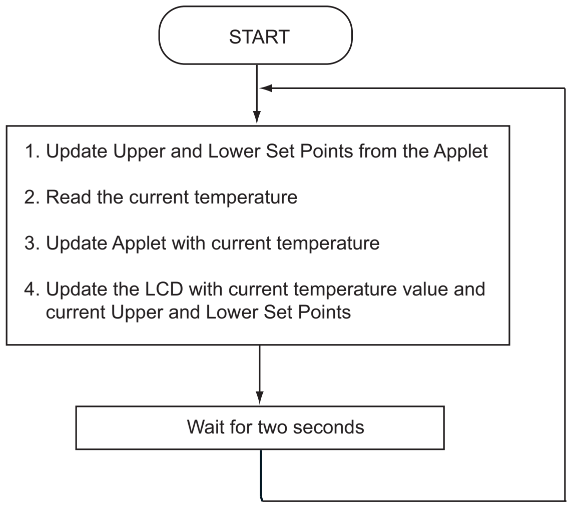

START
1. Update Upper and Lower Set Points from the Applet
2. Read the current temperature
3. Update Applet with current temperature
4. Update the LCD with current temperature value and current Upper and Lower Set Points
Wait for two seconds

Figure 8. Flowchart for the Thermostat Applet and LCD Update

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

As used herein

Document Disclaimer