Application Note

A Web Page—EEPROM Interface Using the eZ80F91 Web Server

AN020903-0808

Abstract

This Application Note describes how to develop a web page interface to transfer data from/to a peripheral hardware device (an I^2C EEPROM in this application) using the web server capabilities offered by the eZ80F91 in conjunction with Zilog TCP/IP (ZTP) software. The web page interface provides a communication interface between a web page and an EEPROM device via the I2C peripheral of the eZ80F91.

Using the HTTP-specific APIs of the ZTP stack and the on-chip I2C peripheral, the data received from the web page is stored in the I2C EEPROM and, upon a request received from a web page, the stored data is retrieved and displayed on the web page.

Notes: 1. *The source code file associated with this Application Note, AN0209-SC01.zip, is available for download at www.zilog.com.*

2. *The code files in this document are intended for use with the ZTP v2.1.0. If you do not intend to develop your application with ZTP v2.1.0, Zilog provides an Application Note about this topic that is intended for use with earlier versions of ZTP.*

Zilog Product Overview

This section contains brief overviews of the Zilog® products used in this Application Note, which includes the award-winning eZ80Acclaim!® microcontroller units (MCU) and the full-feature ZTP software suite.

eZ80Acclaim!® MCU Family Overview

The eZ80Acclaim!® family of MCUs includes Flash and non-Flash products. The Flash-based eZ80Acclaim! MCUs, device numbers eZ80F91, eZ80F92, and eZ80F93, are an exceptional value for you in designing high performance embedded applications. With speeds up to 50 MHz and an on-chip Ethernet MAC (eZ80F91 only), you have the performance necessary to execute complex applications supporting networking functions quickly and efficiently. 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: the eZ80L92 and eZ80190 microprocessors.

ZTP Overview

The ZTP integrates a rich set of networking services with an efficient real-time operating system (RTOS). The operating system is a compact preemptive multitasking, multithreaded kernel with inter-process communications (IPC) support and soft real-time attributes. Table 1 on page 2 lists the standard network protocols implemented as part of the embedded TCP/IP protocol stack in ZTP.

Table 1. Standard Network Protocols in ZTP

HTTP	TFTP	SMTP	Telnet (Client and Server)	IP	PPP	SNTP
DHCP	DNS	TIMEP	SNMP	TCP	UDP	SSL
ICMP	IGMP	ARP	RARP	FTP (Client and Server)		

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.

Discussion

Figure 1 displays the general data communication path using the eZ80F91 Development Kit with the EEPROM device, the Ethernet Smart Cable and an Ethernet Hub. Figure 2 on page 3 displays the hardware setup using a USB Smart Cable. With this hardware, the eZ80F91 is ready to be used as an efficient web server when Zilog's ZTP software is downloaded on it. ZTP provides the software to drive the hardware used for TCP/IP connections. The hardware comprises of a serial (UART) port for PPP connections and the Ethernet Media Access Controller (EMAC) for Ethernet connections.

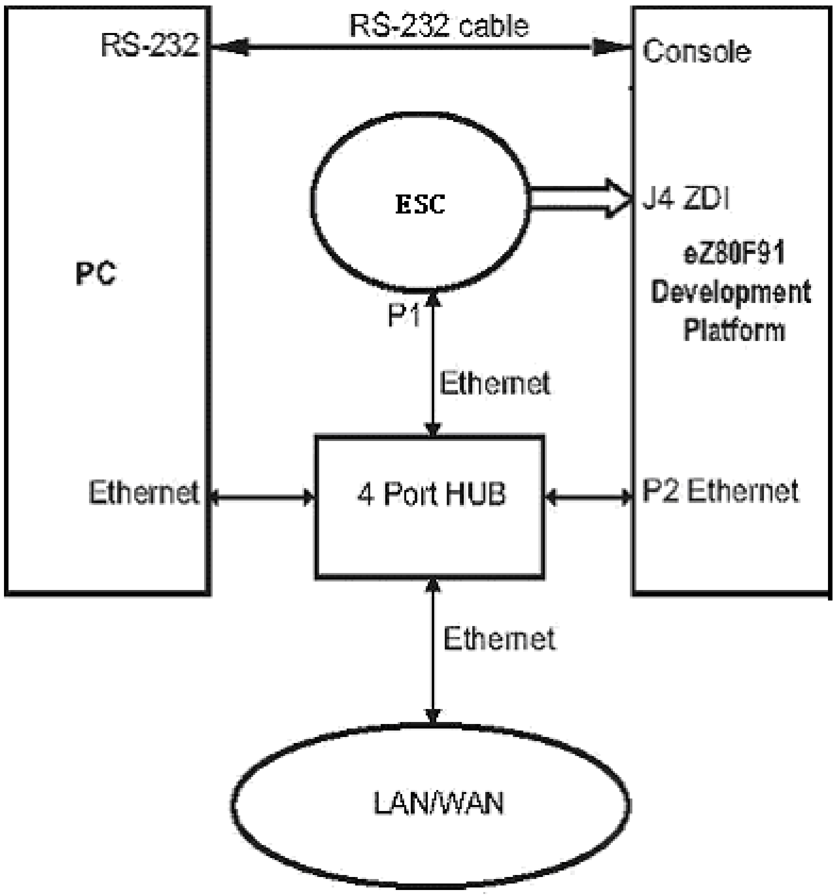


Figure 1. Hardware Setup for the ZTP Application Using Ethernet Smart Cable

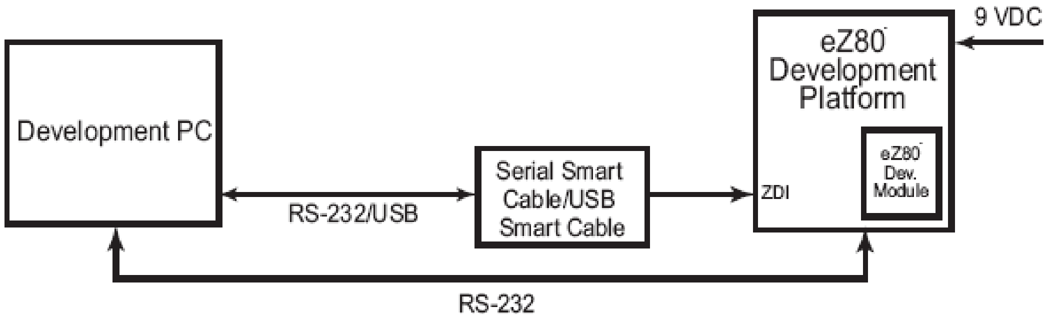


Figure 2. Hardware Setup for the ZTP Applications Using USB Smart Cable

The ZTP contains a set of libraries that implement an embedded TCP/IP stack and includes a preemptive, multi-tasking kernel.

The ZTP APIs allows you to use any member of the eZ80® family of microprocessors/controllers to rapidly develop Internet-ready applications with minimal effort. Because the API is common to all members of the eZ80 family, applications targeting one processor are easily ported to any other eZ80 devices.

ZTP HTTP CGI Functions

Embedded systems typically do not contain a file system. Lack of a file system implies that embedded-systems cannot save CGI scripts as separate `*.cgi` files.

Instead of saving CGI scripts as separate `*.cgi` files, ZTP uses C function calls, collectively called CGI functions. When a CGI function is called, it generates an HTML page that is sent to the browser. It is in these function calls that you write code to read the information sent by a form via a webbrowser. This information is then processed as required by the application. ZTP provides the following CGI functions:

- `int http_output_reply(http_request *request, int reply)`
- `char *http_find_argument (http_request *request, char *arg)`
- `int _http_write (http_request *request, char *buff, int count)`

In each ZTP CGI function, the pointer to the request structure is used to keep the requests from different clients separate.

The function `http_output_reply()` is used to return an acknowledgement to the browser that made the request.

The `http_find_argument()` function is used to extract parameters from the received data in the parsed browser request.

The `macro _http_write ()` is used to return data to the browser that sent the request that invoked the CGI function.

For detailed information about the protocols used in ZTP, refer to *Zilog TCP/IP Software Suite v2.1 Programmer's Guide (RM0041)*, available with the ZTP software.

Developing the Web Page EEPROM Interface Application

This section discusses the I^2C APIs, the EEPROM APIs, the HTML pages, and the CGI functions, and includes a section about how to interface the application-specific files to the standard ZTP.

The I^2C peripheral on the eZ80F91 is used to communicate with the EEPROM. The data received from the web page is stored in the EEPROM and upon request the same information or data is retrieved from the EEPROM and displayed on the web page.

Web Page-EEPROM Interface HTML Files

Figure 3 is a screen shot of the Machine Control Demo web page (`mcd.htm`) that contains two columns: Write Data column and the Read Data column.

The Write Data column allows you to submit data to be programmed into the EEPROM memory, while the Read Data column allows you to read the data obtained from the EEPROM memory.

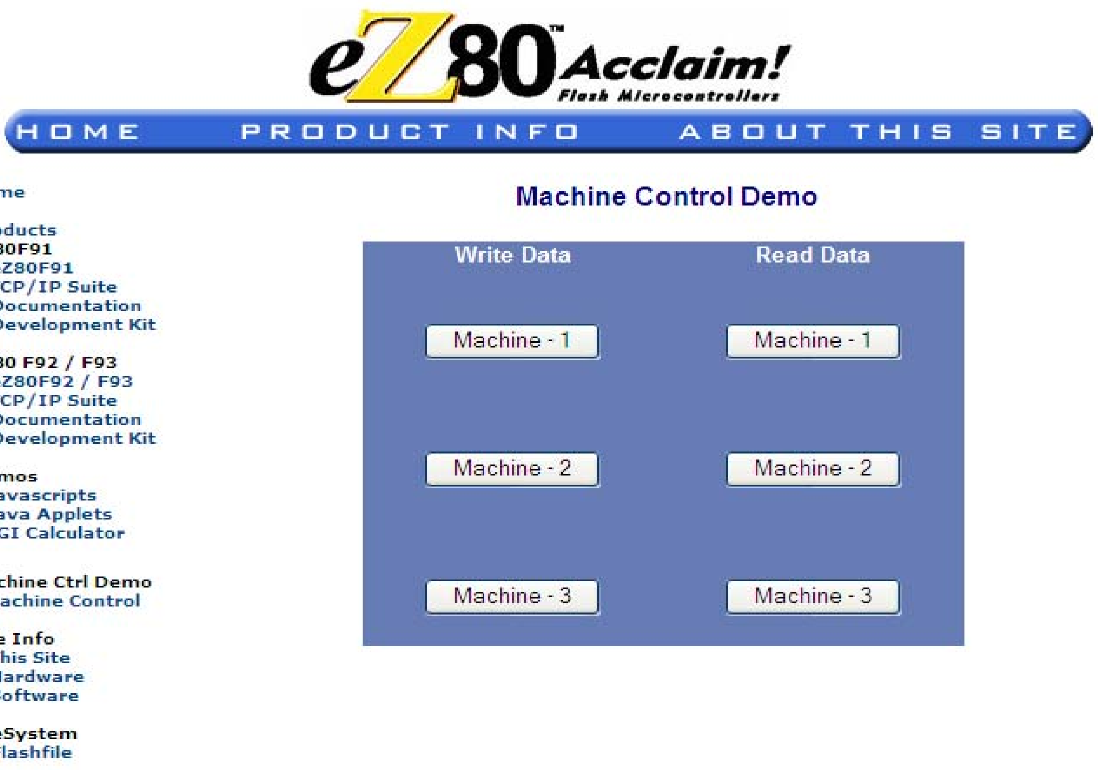


Figure 3. Machine Control Demo Page

The Write Data and the Read Data columns contain three buttons each, enabling you to write to or read from three memory blocks on the EEPROM. These memory blocks are referred to as Machine-1, Machine-2, and Machine-3.

Writing Data to the EEPROM Device

Click a button (Machine-1) in the Write Data column and a pop-up window opens with an HTML submission form that is used to obtain user input. This is an instance of a static web page. To achieve this functionality, a JavaScript method for a pop-up window is called within the `mcd.htm` file. The `wm1()` function is presented below as an example:

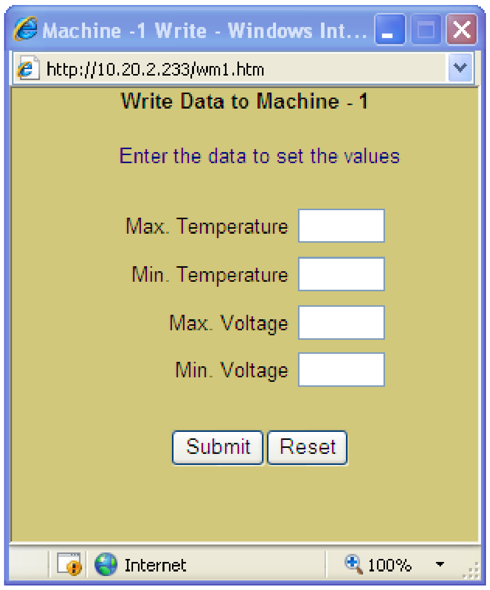
```
function wm1()

{

   popup=widow.open("wm1.htm","","
   height=290,width=300,

   scrollbars=no,screenX=350,
   screenY=250,left=350,top=250");

}
```


Machine -1 Write - Windows Int...
http://10.20.2.233/wm1.htm

Write Data to Machine - 1

Enter the data to set the values

Max. Temperature

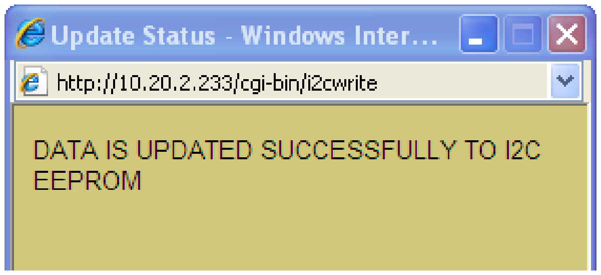
Min. Temperature

Max. Voltage

Min. Voltage

Figure 4. Screenshot of Write Data to Machine

When the form with the user input is submitted, a hidden variable with a value equal to 1 is sent to write the data to the Machine-1 memory block.


Update Status - Windows Inter...
http://10.20.2.233/cgi-bin/i2cwrite

DATA IS UPDATED SUCCESSFULLY TO I2C EEPROM

Figure 5. Pop Up Window When Successfully Updated to EEPROM

In the `mcd.htm` file, three such methods (`wm1()`, `wm2(),and wm3()`) are implemented for the three memory blocks on the EEPROM.

A JavaScript function checks the input fields for the values entered. If the values are not according to specification, then an alert is sent to the Browser window to display an error message. The JavaScript also checks to see that none of the fields are left blank and displays an error if a null value field is found.

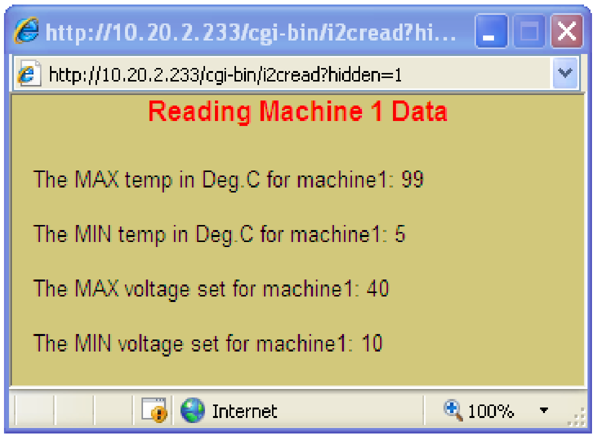
Reading Data from the EEPROM Device

Click a button (Machine-1) in the Read Data column and a pop-up window opens displaying the data retrieved from the EEPROM memory block Machine-1. This is an instance of a dynamic web page. A JavaScript function calls the `i2cread_cgi()` function which is responsible for reading the data from the appropriate memory block (in this case Machine-1 memory block). The `read1()` function is presented below as an example:

```
Function read1()

{

popup = window.open("/cgi-bin
i2cread?hidden=1",        "",
"height=290,width=300,scroll-
bars=no,screenX=350,screenY=250,
left=350,top=250");
}
```


http://10.20.2.233/cgi-bin/i2cread?hi...
http://10.20.2.233/cgi-bin/i2cread?hidden=1
Internet
100%

Reading Machine 1 Data

The MAX temp in Deg.C for machine1: 99

The MIN temp in Deg.C for machine1: 5

The MAX voltage set for machine1: 40

The MIN voltage set for machine1: 10

Figure 6. Screenshot of Read Data to Machine

In the above script, a value identifying the clicked button is sent along with the read request. The value `hidden=1` identifies that a read request was generated for reading the values from the Machine-1 memory block on the EEPROM. The `read2()` and `read3()` functions also call the `i2cread_cgi()` function. The values for `hidden` are 2 and 3, indicating that read requests were generated for reading the values from Machine-2 and Machine-3 memory blocks on the EEPROM.

Application-Specific CGI Functions

In the Web Page-EEPROM Interface application two CGI functions, `i2cread_cgi()` and `i2cwrite_cgi()`, are used. These functions are available in the `i2c_cgi.c` file. These CGI functions call the ZTP HTTP CGI functions to read the user input from the HTML form and to display the information fetched from the EEPROM on the web page. For more information, see ZTP HTTP CGI Functions.

The `i2cwrite_cgi()` function, invoked when a button on the Write Data column is clicked, is structured as follows:

```
int        i2cwrite_cgi(struct
http_request *request)
```

This CGI function calls the ZTP HTTP CGI function, `http_output_reply()`, to send an acknowledgement to the client browser. The `http_find_argument()` function is called to fetch the user input data from the HTML form. This user input data is then transferred over the I2C bus to the EEPROM using the I^{2}C APIs, and the input data is written to the specified memory block in the EEPROM using the EEPROM-specific APIs. The ZTP macro `_http_write` is used to write a success message, which is displayed on the web page.

The `i2c_read_cgi()` function, invoked when a button on the Read Data column is clicked, is structured as follows:

```
int         i2cread_cgi(struct
http_request *request)
```

The `i2c_read_cgi()` function first calls the ZTP HTTP CGI function, `http_output_reply()`, to send an acknowledgement to the client browser. The `http_find_argument()` function is called next to determine from which memory block the data must be read. The EEPROM APIs are then called to read from the specified memory block on the EEPROM. Finally, the ZTP macro `_http_write` is used to write the retrieved data to a pop-up window.

I^{2}C and EEPROM APIs

The I^{2}C Controller on the eZ80F91 controls the write and read operations on the I^{2}C EEPROM.The I^{2}C APIs are not re-entrant. Therefore, while one process is executing, the next process is in queue and executes only after the first completes. Semaphores are used to maintain process synchronization. These semaphores are created and handled within the application-specific CGI functions. The file, `i2c.c`, contains all the APIs specific to I^{2}C peripheral of the eZ80F91.

The `eeprom.c` file contains the APIs to perform the read and write operations on the I^2C EEPROM. These APIs are called within the application-specific CGI functions to write to or read from the EEPROM device.

Adding and Integrating Application Specific Files to ZTP

The Web Page-EEPROM Interface application described in this Application Note, requires the eZ80 Development Platform that contains an EEPROM device and the eZ80F91, with the ZTP.

For executing the application, the files specific for the application must be added and integrated to the ZTP stack. Static web pages (static HTML files) must be uploaded to the external Flash Memory on the eZ80F91 module. This section discusses the details of adding the application-specific files to the ZTP stack and loading the static web pages into the Flash Memory.

The Web Page-EEPROM Interface files that must be loaded to the external Flash Memory to integrate with ZTP are provided in the `AN0209-SC01.zip` file, available for download at www.zilog.com. The application-specific files are of the following types:

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

The ZTP stack is available for download at www.zilog.com and can be downloaded to a PC utilizing a user registration key. ZTP can be installed in any location as specified by you; its default location is C:\Program Files\ZiLOG.

For the version of ZTP and ZDS II used in this application, see Requirements on page 9..

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

1. Download ZTP. Browse to the location where ZTP is downloaded.
2. Download the `AN0209-SC01.zip` file and extract all its contents to a folder on your PC (this folder is referred to as `\\Webpage_Interface` in the rest of the Application Note). The two extracted folders within the WebPage_Interface folder are:

   `\WP_Demo`

   `\WP_Website.Acclaim`
3. Copy all the *.c and *.h files located in the `\WP_Demo` folder to the `..\ZTP\SamplePrograms\ZTPDemo` directory.
4. Launch ZDS II-eZ80Acclaim! and open the `website.zdsproj` file located in the `..\ZTP\SamplePrograms\Website.Acclaim folder.`
5. Open the `website.c` file from ZDS II, and add the following prototype declarations:

   ```
   extern int i2cwrite_cgi (struct
   http_request *request);

   extern int i2cread_cgi (struct
   http_request *request);
   ```
6. The `website.c` file contains an array, `Webpage website[]`, that provides information about the HTML pages. Replace the last line of array,`{0, NULL, NULL, NULL },` with the following code snippet:

   ```
   {HTTP_PAGE_DYNAMIC, "/cgi-bin/
   i2cwrite", "text/html", (struct
   staticpage *) i2cwrite_cgi},
   {HTTP_PAGE_DYNAMIC, "/cgi-bin/
   i2cread", "text/html", (struct
   staticpage *) i2cread_cgi},
   {0, NULL, NULL, NULL}
   ```
7. From the ZDS II-IDE, open the `left.htm` file located in the `\Web Files` folder. Search for **CGI Calculator** and locate the following code snippet:

```
&nbsp;&nbsp; <a href="cgif.htm"
target="_top">CGI

Calculator</a></font><p><font
face="Verdana" size="1">
```

8. To create a link from the **eZ80Acclaim!** home page to the **Machine Control Demo** web page, enter the following HTML code after the code provided in step 7.

```
Machine Ctrl Demo<br>

&nbsp;&nbsp; <a href="mcdf.htm"
target="_top">Machine Control
</a><br><br>
```

9. Compile and build the `website.zdsproj` project.
10. Close the `website.zdsproj` project.
11. In the ZDS II-IDE, open the `ZTPDemo_F91.zdsproj` project located in the `..\ZTP\SamplePrograms\ZTP-Demo` directory.
12. Add all the `*.c` files located in the `..\WebPage_Interface\WP_Demo` folder to the `ZTPDemo_F91.zdsproj` project. To do so, click **Project** and then click **Add** Files. The *.c files to be added are:

```
eeprom.c
i2c_an0209.c
i2c_cgi.c
```

13. Open the `main.c` file of the `ZTPDemo_F91.zdsproj` project and add the following include file:

```
#include "ZSemaphore.h"
#include <eeprom.h>
```

14. Add the following function prototypes and global variables to the `main.c` file:

```
   // global declarations
extern RZK_SEMAPHOREHANDLE_t Sema-
phoreID;
// prototype functions
char load_init_value_eeprom (void);
```

15. Add the following code snippet after the `shell_init(TTYDevID)`; function in the `main.c file`.

```
  SemaphoreID = RZKCreateSema-
phore("Zilog",0,RECV_ORDER_FIFO
);
  if(SemaphoreID  ==  NULL)
printf(" Semaphore NOT cre-
ated");
```

Adding this code creates a semaphore, which is used for controlling the reentrancy issues associated with the I^2C APIs.

16. At the end of the `INT16 ZTPAppEntry (void)` function, add the following function call just before the `return(OK)` statement.

```
load_init_value_eeprom ();
```

17. Add the following function definition after the `main()` function in the `main.c` file.

```
//FunctionName
//load_init_value_eeprom()

char load_init_value_eeprom()
{

unsigned char load;
unsigned char
init_value[]={"0000000000000000
000000000000000000000"};
unsigned char *ptr1 = (unsigned
char *)0x0000;
unsigned char *ptr2 = (unsigned
char *)0x0100;
unsigned char *ptr3 = (unsigned
char *)0x0200;
EEPROM_Open();
EEPROM_Write( ptr1, (unsigned
char *)init_value,40);
EEPROM_Write( ptr2, (unsigned
char *)init_value,40);
EEPROM_Write( ptr3, (unsigned
char *)init_value,40);
EEPROM_Close();
Return 0;
}
```

18. Open the `emac_conf.c` file and change the default MAC address (provided by ZTP) such that each eZ80 Development Platform on the Local Area Network (LAN) contains a unique MAC address. For example:

    ```
    INT8 f91_mac_addr[ETHPKT_ALEN] =
    {0x00,0x90,0x23,0x00,0x0F,0x91};
    ```

    In the six byte MAC address listed above, the first three bytes should not be modified; the last three bytes can be used to assign a unique MAC address to the eZ80 Development Platform.
19. Open the `ZTPConfig.c` file. The Dynamic Host Configuration Protocol (DHCP) is disabled for this application, therefore, ensure that

    `b_use_dhcp = FALSE.`
20. Search the following structure definition in the `ZTPConfig.c` file.

    ```
    struct If ifTbl[MAX_NO_IF]= {
    /*interface 0 -> Ethernet Configura-
    tion*/
    {
    &usrDevBlk[0],/* Control
                     block for this device*/
    ETH,          /* interface type */
    ETH_MTU,      /* MTU */
    ETH_100,      /* Speed ETH_100,
                  ETH_10,ETH_AUTOSENSE */
    "192.168.1.103",/*Default IP address
    */
    "192.168.1.1", /* Default Gateway */
    0xffffff00UL /* Default Subnet
    Mask*/
    },
    ```

    The structures listed above contain network parameters and settings (in the four-octet dotted decimal format) specific to the local area network at Zilog, as default.

Note: *Modify the above structure definition with appropriate IP addresses within your LAN. For details about modifying the structure definition, refer to ZTP 2.1 documents listed in* References on page 13.

21. Take note of the following
    a. In `main.c` file uncomment the `Initialize_FileSystem();`
    b. In `ZTPConfig.c` file make `"g_ShellLoginReqd = FALSE"`
22. Compile and build the `ZTPDemo_F91.zdsproj project`.

Demonstration

This section contains the requirements and instructions to set up the Machine Control Interface Demo and run it.

Requirements

The requirements are classified as hardware and software.

The hardware requirements include:

- eZ80F91 Development Kit
- eZ80Acclaim!® Development board
- PC with an Internet browser with the Java Virtual Machine.

The software requirements include:

- ZiLOG Developer Studio II-IDE for eZ80Acclaim! 4.11.0 (ZDSII 4.11.0)
- ZiLOG's TCP/IP stack (ZTP 2.1)

Setup

The basic setup to assemble the Machine Control Interface demo is displayed in Figure 1 on page 2. This setup illustrates the connections between the PC, Smart Cable, LAN/WAN/Internet, and the eZ80F91 Development Kit.

Settings

HyperTerminal Settings

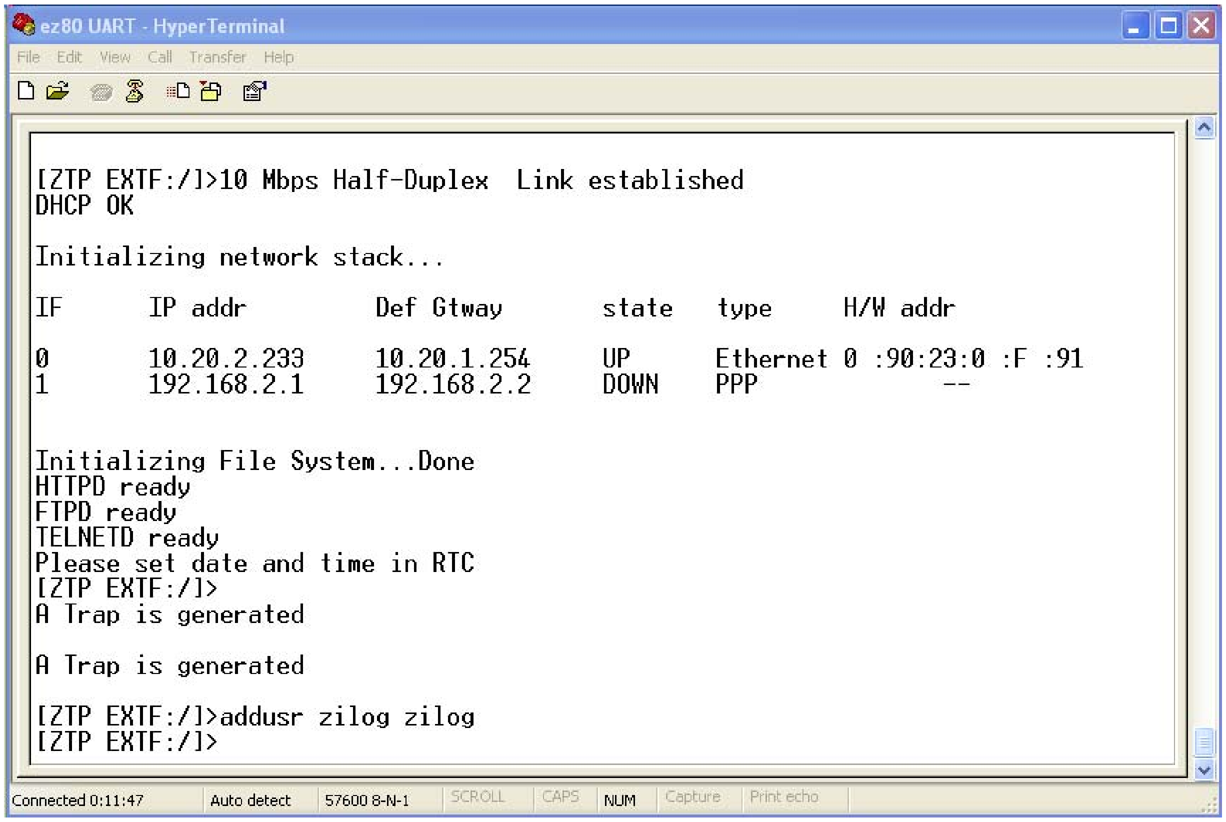
Following are the HyperTerminal Settings:

- Set HyperTerminal to 57.6 Kbps Baud and 8-N-1, with no flow control

Procedure for setting up the Machine Control Interface

The procedure to build and run the Machine Control Demo is described in this section.

1. Launch a HyperTerminal application and configure the HyperTerminal application to the following settings: 57600 bps baud, 8-N-1, with no flow control.
2. From the ZDS II-IDE, download the `ZTPDemo_F91.lod` (Flash or RAM) output file to the target platform. To execute the program, click the GO icon in the ZDS II-IDE. Observe the ZTP shell prompt [ZTP EXTF:/]> in the HyperTerminal application.
3. Create a user on the eZ80 target board, where the ZTP stack is executing through the shell prompt. Utilize the addusr command to create a user and password if required. Creating a user is a must to store html pages. Different options are displayed by typing the ? symbol, in the shell.



```
[ZTP EXTF:/]>10 Mbps Half-Duplex  Link established
DHCP OK

Initializing network stack...

IF      IP addr         Def Gtway       state   type     H/W addr

0       10.20.2.233     10.20.1.254     UP      Ethernet 0 :90:23:0 :F :91
1       192.168.2.1     192.168.2.2     DOWN    PPP             --


Initializing File System...Done
HTTPD ready
FTPD ready
TELNETD ready
Please set date and time in RTC
[ZTP EXTF:/]>
A Trap is generated

A Trap is generated

[ZTP EXTF:/]>addusr zilog zilog
[ZTP EXTF:/]>
```

Figure 7. Screen Shot Displaying the ZTP addusr Command

4. All html pages must be stored in the external Flash Memory, on the eZ80 target board. Launch the command prompt in the PC and select the directory from which the html files are to be copied (in this case, <web files installed folder>\WP_Website.Acclaim\). Use FTP to transfer the html pages from the PC to the eZ80 target board.

   The following example describes the transfer of html pages from the PC to the eZ80 target board, using FTP. `At the`

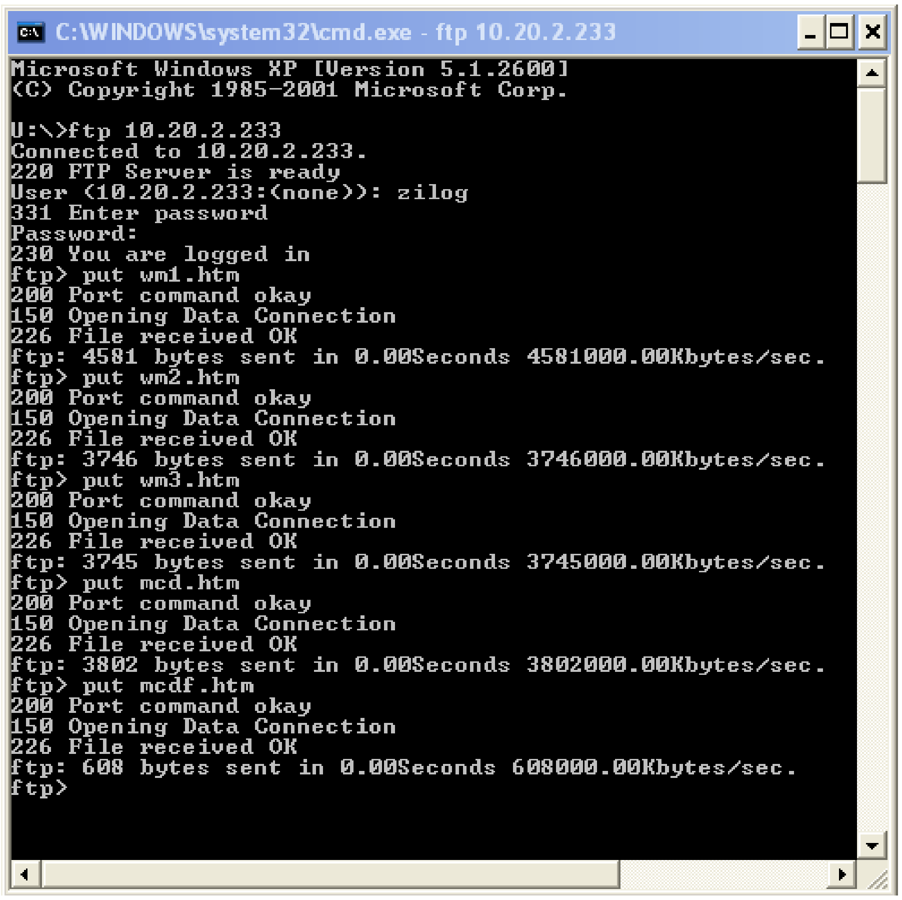
   `<web files installed folder>\WP_Website.Acclaim\PC` command prompt, type the IP address of the target platform. Enter the user name and password when prompted. Use the put function to load the following files:

   a. `Wm1.htm`
   b. `Wm2.htm`
   c. `Wm3.htm`
   d. `Mcd.htm`
   e. `Mcdf.htm`

These files are stored in the root directory of the target file system. If you want to store these files in any other directory, then the path has to be defined accordingly. By default, the array in the `ZTPConfig.c` file is set to the root directory as follows:

```
INT8* httppath = "/" ;
```

Figure 8 displays the command prompt view of transferring files using FTP. (In this example the files are located at the root directory of U:\ )


C:\WINDOWS\system32\cmd.exe - ftp 10.20.2.233

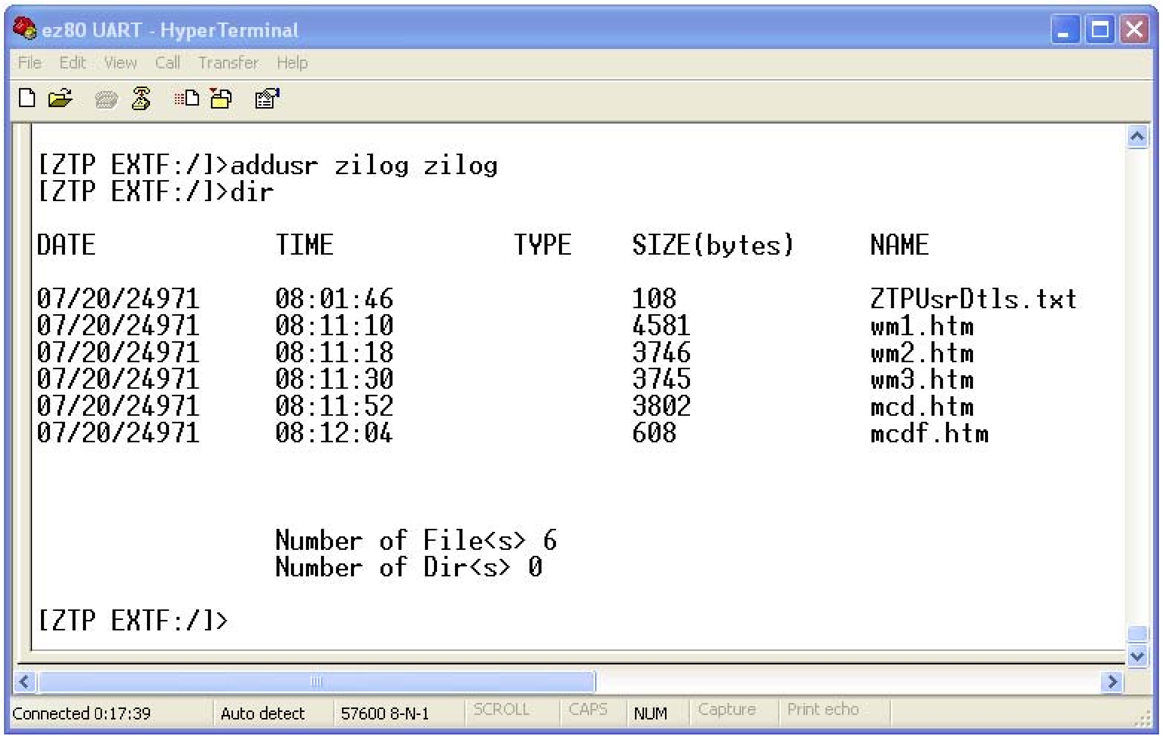
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

U:\>ftp 10.20.2.233
Connected to 10.20.2.233.
220 FTP Server is ready
User (10.20.2.233:(none)): zilog
331 Enter password
Password:
230 You are logged in
ftp> put wm1.htm
200 Port command okay
150 Opening Data Connection
226 File received OK
ftp: 4581 bytes sent in 0.00Seconds 4581000.00Kbytes/sec.
ftp> put wm2.htm
200 Port command okay
150 Opening Data Connection
226 File received OK
ftp: 3746 bytes sent in 0.00Seconds 3746000.00Kbytes/sec.
ftp> put wm3.htm
200 Port command okay
150 Opening Data Connection
226 File received OK
ftp: 3745 bytes sent in 0.00Seconds 3745000.00Kbytes/sec.
ftp> put mcd.htm
200 Port command okay
150 Opening Data Connection
226 File received OK
ftp: 3802 bytes sent in 0.00Seconds 3802000.00Kbytes/sec.
ftp> put mcdf.htm
200 Port command okay
150 Opening Data Connection
226 File received OK
ftp: 608 bytes sent in 0.00Seconds 608000.00Kbytes/sec.
ftp>
```

Figure 8. Command Prompt View of File Transfer Using FTP

5. In the HyperTerminal application, type `dir` to ensure that the files are copied properly.

Figure 9 displays the files stored in Flash after file transfer using FTP.



```
[ZTP EXTF:/]>addusr zilog zilog
[ZTP EXTF:/]>dir

DATE              TIME              TYPE    SIZE(bytes)     NAME

07/20/24971       08:01:46                  108             ZTPUsrDtls.txt
07/20/24971       08:11:10                  4581            wm1.htm
07/20/24971       08:11:18                  3746            wm2.htm
07/20/24971       08:11:30                  3745            wm3.htm
07/20/24971       08:11:52                  3802            mcd.htm
07/20/24971       08:12:04                  608             mcdf.htm



                  Number of File<s> 6
                  Number of Dir<s> 0

[ZTP EXTF:/]>
```

Figure 9. Screen Shot Displaying the Files Stored in Flash after File Transfer Using FTP

Running the Web Page-EEPROM Interface Demo

Follow the steps below to run the Demo:

1. Launch the Internet Browser on the PC. Enter the IP address specified in `ZTPConfig.c`. or indicated in the HyperTerminal window. The `Index.html` page is displayed.
2. Click **Machine Ctrl Demo** link located at the left pane. The **Machine Control Demo** page is displayed.
3. Click **Machine-1** button under the **Write Data** column. The **Write data** pop-up window opens displaying a submission form. The submission form consists of the **Max Temperature**, **Min Temperature, Max Voltage, and Min Voltage** fields.

Note: *Enter values between 0 and 99.99. Do not enter characters; only integers and floating-point values are accepted. The value field is limited to five integers with a dot in between to represent the floating-point value. Entries not complying with these specifications cause an error message to be displayed.*

4. Set the maximum and minimum temperature and voltage values for Machine-1. Click **Submit** button to submit the data. A status pop-up window displays that the data is successfully stored in the EEPROM.
5. Repeat steps 3 and 4 using the **Machine-2** and **Machine-3** buttons under the **Write Data** column to set the values for those machines.

6. Go back to the **Machine Control Demo** page by clicking **Machine Ctrl Demo** link in the left pane.
7. Click **Machine-1** button under the **Read Data** column. The **Read data** pop-up window displays the values set for Machine-1.
8. To read the values set for Machine-2 and Machine-3, click **Machine-2** and **Machine-3** buttons under the **Read Data** column.

Observations

The values set for the machines were successfully written to the EEPROM device via the web page interface and I^2C. This is confirmed by comparing the data read from the EEPROM device.

Summary

This Application Note demonstrates a method of capturing user input data from a web page and storing it on an external device, and further displaying the stored data on the web page upon request. This application can be further extended and used as an industrial web server, where the web server can be connected to different machines or control equipment that can be monitored and controlled through the intranet or the Internet.

References

The documents associated with eZ80®, and eZ80Acclaim!® family of products are listed below:

- eZ80® CPU User Manual (UM0077)
- eZ80F91 Development Kit User Manual (UM0142)
- eZ80F91 Product Specification (PS0192)
- eZ80F91 Module Product Specification (PS0193)
- Zilog TCP/IP Stack API Reference Manual (RM0040)
- Zilog TCP/IP Software Suite Programmer's Guide (RM0041)
- Zilog TCP/IP Software Suite Quick Start Guide (QS0049)
- Zilog Developer Studio II-eZ80Acclaim!® User Manual (UM0144)

Warning: DO NOT USE IN LIFE SUPPORT