HTTP Virtual Peripheral Implementation

SCENIX

Application Note 25
Christopher Waters
December 1999
Patents Pending

1.0 Introduction

This application note describes an implementation of the Hypertext Transfer Protocol (HTTP) for the Scenix SX communications controller.

HTTP is the protocol used by the world-wide-web (WWW). When a user navigates to a page with their web browser an HTTP request is sent from the browser to the HTTP server the web page resides on. The server responds with the resource requested.

This Virtual Peripheral (VP) is an HTTP server implementation. This means that the SX can serve web pages, images, Java™applets, PDF documents, or any other type of file. The WWW provides an easy to use, graphically oriented interface. It is a low resource method of adding a complete graphical user interface (GUI) to the SX. The beauty is that while the formatting commands are sent from the SX, the actual layout and user interface is done by the web browser.

The SX uses an external EEPROM to store the resources that it serves. This way, the total size of the resources is limited only by the size of the EEPROM.

The HTTP VP requires the transmission control protocol (TCP) and TCP/IP stack described in application notes AN27 (TCP Virtual peripheral Implementation) and AN23 (UDP/PPP Virtual Peripheral Implementation).

2.0 HTTP Example

This example shows how HTTP is normally used. The client sends a request for the resource named index.html to the SX HTTP server. The first word, GET, indicates that this request is to get the resource. The SX only interprets the first line of an HTTP request. Subsequent lines give information that may be used by the web server to tailor its response to the client.

The server reply packet is formulated on the SX. Since all of the information, except the date, is constant the HTTP reply header is stored in the EEPROM along with the resource. After the header, a blank line indicates the start of the resource. The data is not encoded in any special way.

Client packet:

```
GET /index.html HTTP/1.0
Host: www.celsius.co.nz
Accept: image/gif, image/jpeg, */*
User-Agent: Lynx/2.6
```

Server reply:

```
HTTP/1.0 200 Document follows
Date: Fri, 9 Jul 1999 09:17:32 GMT
Server: SX-NET/1.0
Last-modified: Fri, 9 Jul 1999 09:15:11 GMT
Content-type: text/html
Content-length: 853

<HTML>…
```

3.0 Implementation

So that the data can be transferred in segments larger than the available RAM, the HTTP implementation uses the event driven architecture described in application note AN23.

HTTP uses a uniform resource identifier (URI) to specify the resource that should be returned. Normally the URI is structured like a file path with directories specifying the location and suffix indicating the file type. Implementing a file system on the SX would require both significant code space and the need to access the EEPROM to search for the file name. The HTTP VP uses a clever hashing scheme to avoid these problems while still allowing full URIs to be used.

When a request is received an 8-bit hash is computed over the URI (hash function is a simple 8-bit sum of the characters in the string). The hash value is then multiplied by two and used as a lookup into a 512 byte table of 16-bit file offsets (called the index block). Hashing collisions are ignored. Using this method avoids any need to do string comparisons, or to store the URIs on the server at all. When the lookup table is created the user is invited to change the URIs of any resources which have a hash collision. It is possible that a GET request might contain a

URI which does not exist on the web-server but which hashes to the same value as a resource which does exist. In this situation the incorrect resource will be returned. However, since most GET requests are generated as the result of hyperlinks from other resources it is very unlikely that an erroneous URI will be generated. The problem of returning the wrong page is no different than the possibility of a user typing a garbage URL into their browser and having it bring up a real page (although a hash collision does have a higher probability of occurrence). One limitation of the table lookup scheme is that the web-server is limited to 256 resources. In the types of applications the SX web-server is designed for this is not a significant limitation.

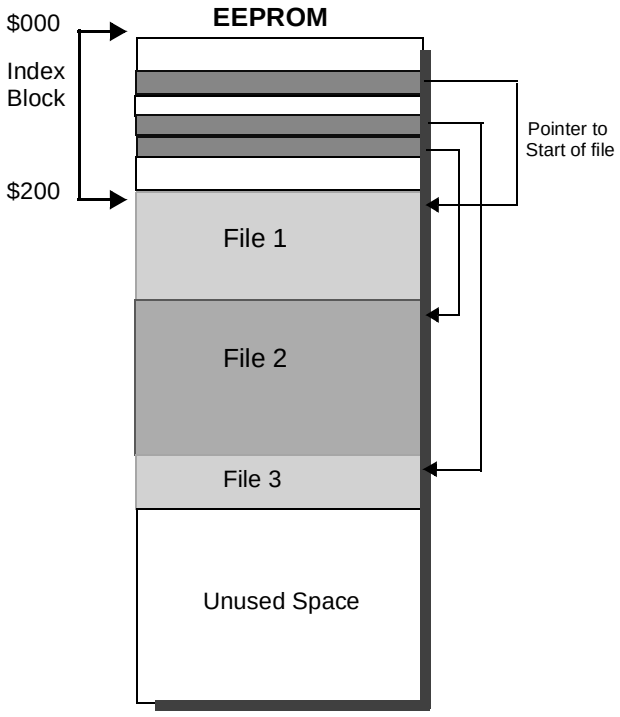

Figure 3-1. File System Implementation

4.0 Updating the Server EEPROM

Before the web-server can be used, the resources need to be downloaded to the EEPROM. This requires programming the SX with different firmware in the file E2File.src. This program can communicate with a PC over a serial port to accept a file to write into the EEPROM. On the PC side the program E2Send.exe is used to create the EEPROM file system. This program will load a directory structure from disk and then download it to the SX. The download uses the debug port on the reference hardware.

The following steps should be followed:

1. Connect the debug port of the reference hardware to COM2 on a PC.
2. Run the SX firmware in the file E2File.src
3. Create the web-server directory structure in the directory c:\temp\html\ on the PC. All files in this directory will be downloaded to the SX's EEPROM.
4. Run the E2Send.exe program. Click 'Find Files' to load the files from disk. If no file names appear in the box then the directory c:\temp\html\ was not found. The number to the left of each file is its hash. Check that no two files have the same hash. If there are two files with the same hash then rename one file and repeat this step.
5. Click 'Build Data', then click 'Send Data'. A window should appear saying 'Found SX…'. If no window appears then either the SX is not running the correct firmware or the serial cable is not connected.
6. The download will proceed. When it is finished the EEPROM is updated with the new directory structure.

5.0 Dynamic Pages

To show changing information web-servers can normally generate the HTML pages they serve on the fly. Creating dynamic HTML in the SX web-server would be possible, however each application has different requirements and writing the HTML generation code for byte-at-time processing can be difficult. Fortunately, for most web applications there is an alternative solution using Java™ applets. A Java™ applet is a program written in Java™ which executes in a browser window. Since it executes on the browser there are no limitations on memory usage or program structure. The SX web-server can serve any type of file, including Java™ applets.

The other feature of dynamic web pages is the HTTP POST method. This is the feature used by HTML forms to send information back to the server. It would be possible to implement the HTTP POST method in the SX web-server, but interpreting the information that is sent back, which are usually strings, would be difficult in an SX. Java™ applets also provide a solution to this problem by allowing pre-processing to take place on the browser.

To create dynamic pages with feedback an applet is created which is stored on the SX web-server. When the page containing the applet is viewer the browser fetches the applet class file from the server and executes it. To get information from the SX or to send back commands

the applet can open a TCP connection or send UDP packets. These packets can be structured for easy interpretation on the SX.

5.1 JAVA™/Sprinkler Demo

As a demonstration of using a Java™ applet a sprinkler control program has been created. The web page, sprinkler.html, has an applet embedded in it. The applet creates a simple form on the web page with checkboxes to turn different zones on or off and a menu to select the length of time each zone is on. When the applet starts executing it sends a UDP packet to the SX to get the current sprinkler settings. When the user clicks the 'Update' button another packet is sent containing the updated settings.

For the demo to work several conditions need to be met:

· Java™ must be enabled in the browser

· The SX must be listed by the DNS as 'sx'. (Done under windows by adding an entry to the \windows\hosts or \ winnt\system32\hosts files.)

· The applet will only work when loaded from the SX. Applets have the restriction that they can only communicate with the same server that they were loaded from.

LIT#:SXL-AN25-04

For the latest contact and support information on SX devices, please visit the Scenix Semiconductor website at www.scenix.com. The site contains technical literature, local sales contacts, tech support and many other features.

SCENIX

1330 Charleston Road
Mountain View, CA 94043
Tel: (650) 210-1500
Fax: (650) 210-8715
Web Site: www.scenix.com