



The MatchPort AR Missing Tutorial

Scott Hendrickson

September 29, 2008 - Version 1.1

Table of Contents

The MatchPort AR Missing Tutorial.....	1
Why an unofficial MatchPort AR Tutorial?.....	2
What does the MatchPort AR do?.....	2
MatchPort AR Configuration.....	2
Is the MatchPort AR a Client or a Server?.....	3
Example: Retrieving the Google homepage via HTTP.....	3
Example: Passing the IP address to retrieve the Google homepage via HTTP.....	6
MatchPort AR Modem Commands.....	7
How do I get my Web pages onto the MatchPort AR?.....	8
Example: FTP new pages to the MatchPort AR.....	9
How do Web pages access the serial ports?.....	10
Example: Text transferred via TCP tunnel between Serial Line 1 and Java Applet.....	11
Java Applet Running in Browser—Tunnel to Serial Line 1.....	12
RealTerm Serial Line 1—Tunnel To Java Applet.....	13
Configuring SSH Keys.....	13

Why an unofficial MatchPort AR Tutorial?

Because there doesn't seem to be an official one.

I bought a MatchPort AR (<http://www.lantronix.com/device-networking/embedded-device-servers/matchport-ar.html>) module a few weeks ago from one of Lantronix's two recommended online vendors in the US, GridConnect Networking Products (<http://www.gridconnect.com/lantronix.html>).

I chose the MatchPort AR because, with the same experience, hardware, and nearly the same software configuration, I should be able to upgrade to a wireless solution with the MatchPort b/g Pro. (The MatchPort b/g uses a different OS—I don't know if everything here applies to the MatchPort b/g. If anyone tries it, please let me know.)

I wanted to get up to speed quickly, so I purchased the demo kit, thinking that would include all the software and hardware I would need to understand and learn how to use the MAR in a project of my own. This was more or less the case, but getting up to speed was much slower than I expected.

I thought I would be up and running demo apps in an afternoon and off to my first project early the next morning. Instead, I spent hours understanding the paradigm of intended use assumed by the MAR, hunting down example code, working out details and trying to understand the MAR's somewhat convoluted documentation.

I hope this tutorial gives the first-time embedded TCP/IP/HTTP server users a fast and straightforward introduction to the MAR. And maybe I can save more experienced users the hassle of tracking down the details of how the MAR can be coaxed into becoming a working embedded network component.

What does the MatchPort AR do?

It does a lot. But the basic ideas are pretty simple: It has an Ethernet connection at one end and 2 serial ports at the other:

1. Web Server. On the Ethernet end, the MatchPort AR can act as a web server, serving up HTML pages with Javascript/AJAX capabilities.
2. Tunnel any text you like from either of 2 serial ports to a range of Internet protocols over Ethernet. For example, the MAR can be configured to send text from a serial port to TCP packets, UDP packets, SSH sessions, Telnet sessions...

Also, the MAR can be hardware interfaced. Beside 2 TTL level serial ports, the MAR has a handful of control pins that can be used to configure interfacing, send emails, output specified logic levels, or read logic levels.

MatchPort AR Configuration

Configuration is extensively described in the included documentation included with the MatchPort AR. You can download it from the Lantronix site at <http://www.lantronix.com/support/documentation.html>. This tutorial will help you understand *why* you will want to configure the MAR and maybe give you some ideas on how the MAR will can solve your need for embedded networking

There is also an SDK that enables a developer to alter or extend the firmware of the MatchPort AR. Nothing in this tutorial requires the SDK.

The MAR can be configured from either end and over many of the protocols. You can configure the MAR via the Web (my preferred), SSH, Telnet, and menu of options via the serial with a command line interface, the CLI. My MAR does not get along with Firefox 3.0, so I have to fall back to Microsoft Internet Explorer.

Follow the instructions in the documentation for quick setup to get an IP for the MAR and to get access to the Web configuration application.

Is the MatchPort AR a Client or a Server?

Either—sort of three choices, really.

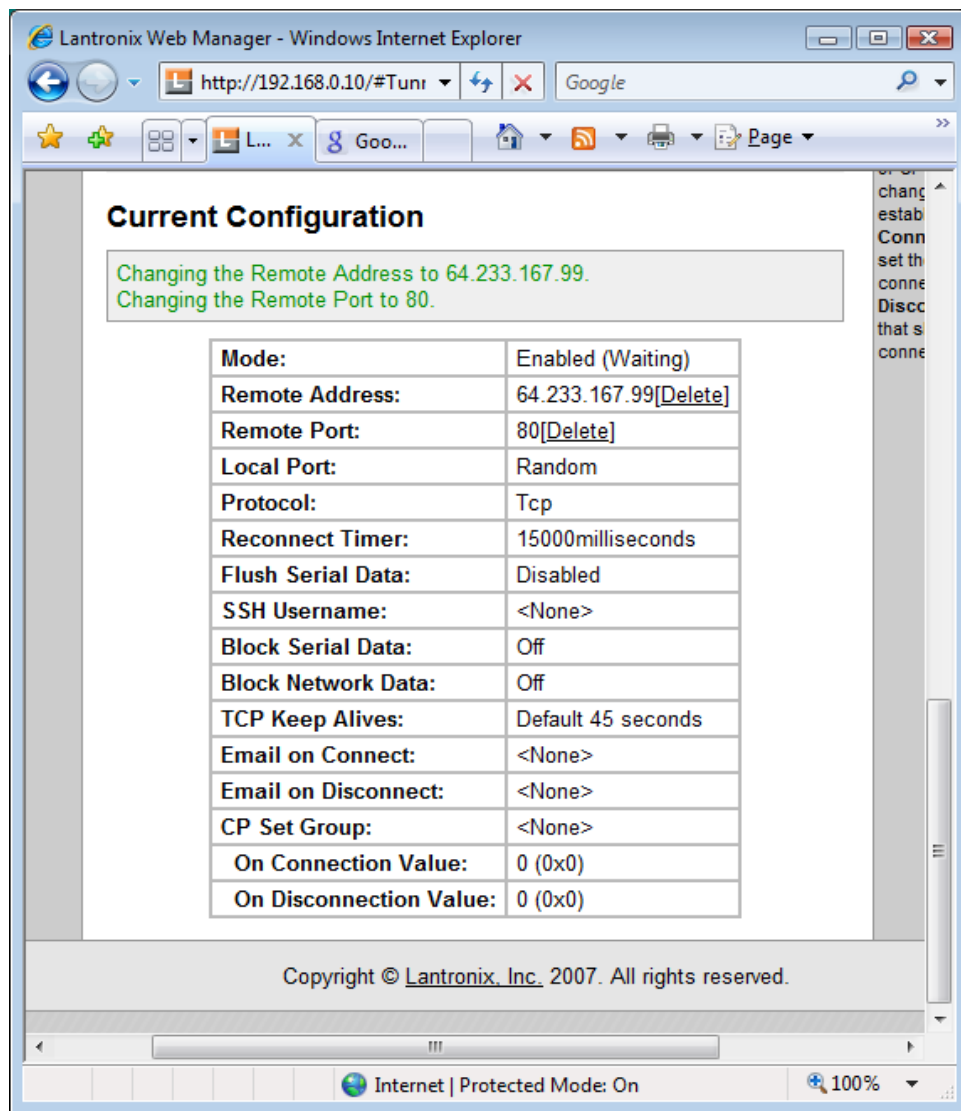
On the Ethernet end, the MAR plays server for SSH, Telnet, HTTP, FTP and RSS. On the Serial lines, the MAR can be the CLI server for configuring the MAR.

When the MAR is tunneling text in either direction from TCP packets or SSH sessions on the Ethernet end and the Serial Lines, the MAR itself is transparent—a tunnel rather than a client or a server.

And the MAR can help your project play the client role too. Here are two examples:

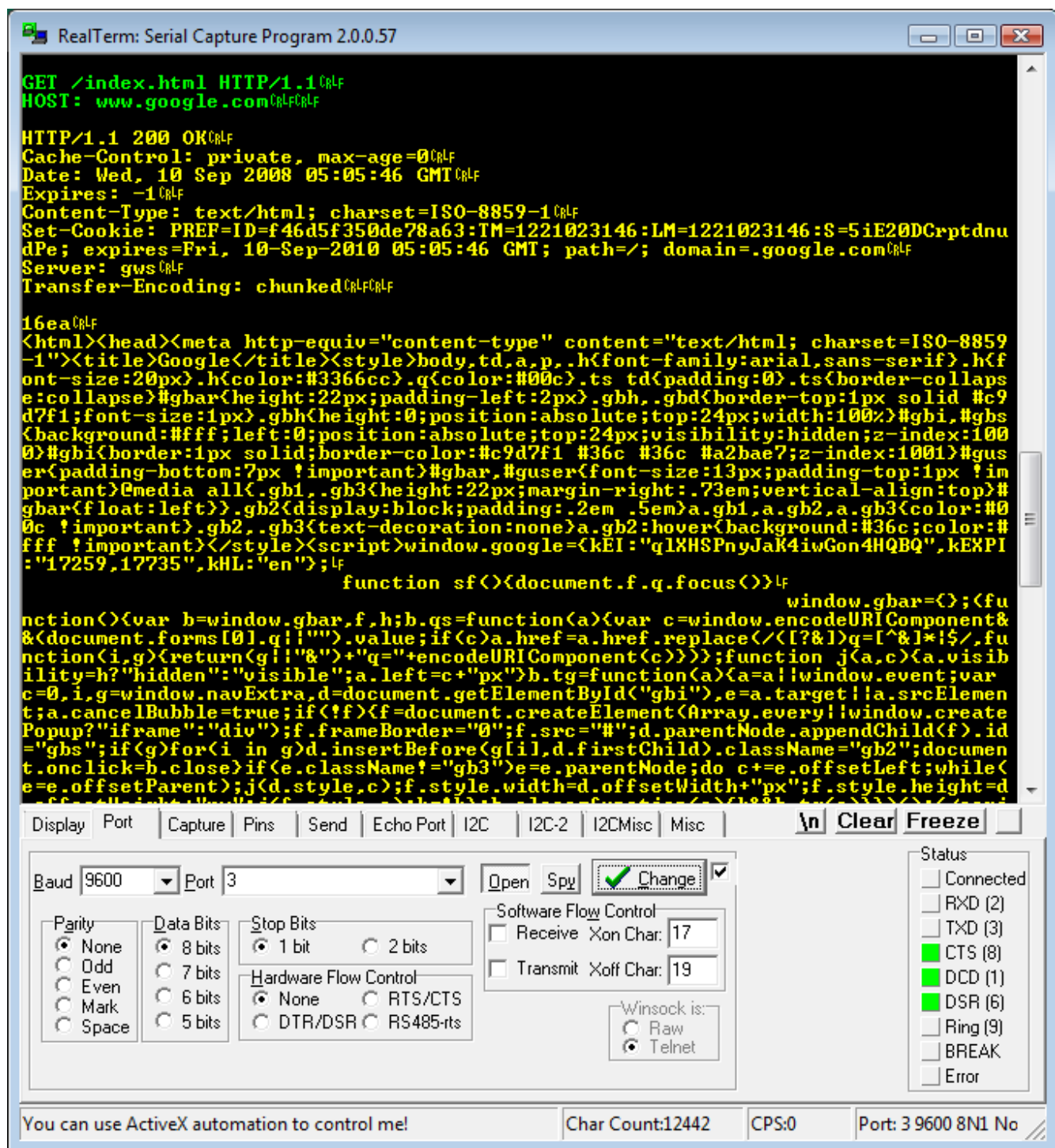
Example: Retrieving the Google homepage via HTTP

First, I configured Serial Line 1 to Tunnel Serial 1 with Command Mode Disabled. In the Tunnel menu, set Tunnel 1 Connect Mode to Enabled, TCP and enter the IP address of Google (64.233.167.99), accessing port 80 for the Web server. This is a semi-static configuration for accessing Google. See the screen shot. (Below, I will show how to pass the IP address from the Serial Line.)



The HTTP request of the Google homepage is made by sending the text below over the Serial Line. The MAR wraps it in a TCP packet and the request is sent off the Google servers. Be sure that your terminal program has the same settings as your Serial Line/Configuration setup. I use both RealTerm (<http://realterm.sourceforge.net/>) and PuTTY (<http://www.chiark.greenend.org.uk/~sgtatham/putty/>) because each does some things easier than the other, but neither of them does it all.

```
GET /index.html HTTP/1.1
HOST: www.google.com
```



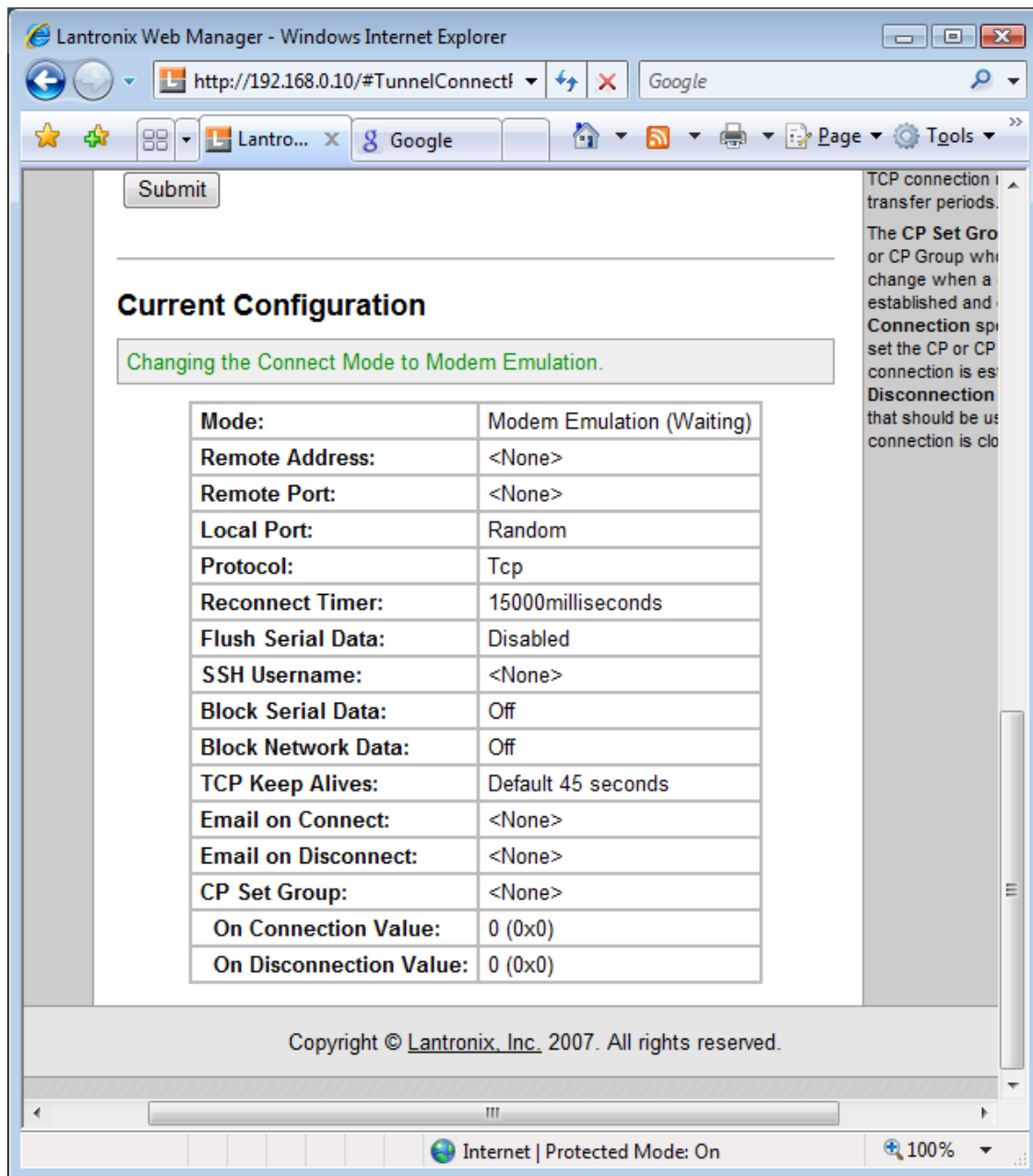
Be sure to follow the command with two CR/LFs. In the screen shot, you can see the HTTP header followed by the HTML content of the page. In an embedded application, this text might be parsed by the micro controller connected to Serial Line 1, for example.

I used RealTerm because you can copy the text from a text editor and paste to the port (right click menu). This is nice because there is not backspace and, when there is no local text echo, you can't see

what you have already typed.

Example: Passing the IP address to retrieve the Google homepage via HTTP

Clear the Remote Address and Remote Port fields in the Tunnel 1|Connect Mode menu. Change the Mode from Enabled to Modem Emulation.



The serial terminal window connected to your serial port now answers the command AT with “OK”. Below are the MatchPort AR Modem Commands, not found in the printed documentation.

MatchPort AR Modem Commands

+++	Switches to command mode if entered from serial port during connection.
AT?	Help. Displays this table.
ATD	Sets up a TCP connection with default connect mode IP and port.
ATD<IP address>:<port>	Sets up a TCP connection. A value of 0 begins a command line interface session
ATDP<IP address>:<port>	See ATD.
ATDT<IP address>:<port>	See ATD.
ATO	Switches to data mode if connection still exists. Reverse of '+++'.
ATH	Disconnects the network session.
ATI	Displays modem information.
ATEn	Switches echo in command mode (n=0: off, n=1: on).
ATQn	Quiet mode (n=0: enable results code, n=1: disable results code.)
ATVn	Verbose mode (n=0: numeric result codes, n=1: text result codes.)
ATXn	Command does nothing and returns OK status.
ATUn	Accept unknown commands. (n=0: off, n=1: on).
AT&V	Display current and saved settings
AT&F	Reset saved settings in NVR to factory defaults.
AT&W	Save active settings to NVR.
ATZ	Restores the current state from the setup settings.
ATS0=n	Accept incoming connection. (n=0: disable, n=1: connect automatically, n=2+: connect with ATA command).
ATA	Answer incoming connection (if ATS0=2 or greater).
A/	Repeat last valid command.

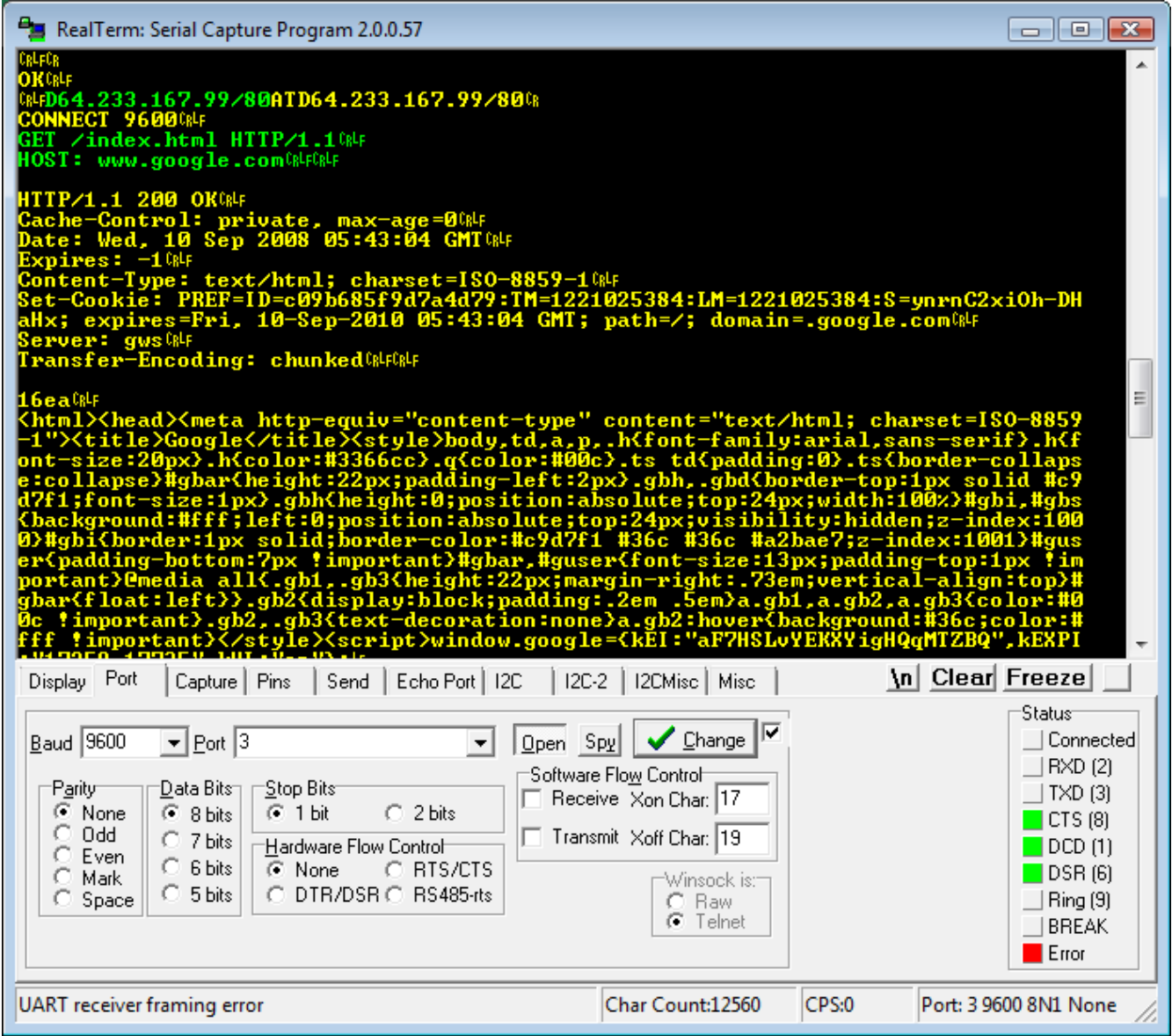
Now, to make the connection and the HTTP request:

First connect to the Google server with the ATD command. The “/80” shown in the command example below is the notation for telling the client to connect to the remote server on port 80. In the screen shot from the serial RealTerm session (see below) the command is echoed by the modem emulation program (responses in Yellow) so it appears twice before the CR/LF.

```
ATD64.233.167.99/80
```

Wait for the CONNET 9600 response. Then make the HTTP request as we did before. I copied and pasted from a text editor using the right-click menu in RealTerm. If you take too long to copy or type the request, the connection may time out.

```
GET /index.html HTTP/1.1
HOST: www.google.com
```



The screenshot shows the RealTerm Serial Capture Program interface. The main window displays a serial capture log with the following text:

```
CR LF
OK CR LF
CR LF D64.233.167.99/80 AT D64.233.167.99/80 CR
CONNECT 9600 CR LF
GET /index.html HTTP/1.1 CR LF
HOST: www.google.com CR LF CR LF

HTTP/1.1 200 OK CR LF
Cache-Control: private, max-age=0 CR LF
Date: Wed, 10 Sep 2008 05:43:04 GMT CR LF
Expires: -1 CR LF
Content-Type: text/html; charset=ISO-8859-1 CR LF
Set-Cookie: PREF=ID=c09b685f9d7a4d79:IM=1221025384:LM=1221025384:S=ynrnC2xi0h-DH
aHx; expires=Fri, 10-Sep-2010 05:43:04 GMT; path=/; domain=.google.com CR LF
Server: gws CR LF
Transfer-Encoding: chunked CR LF CR LF

16ea CR LF
<html><head><meta http-equiv="content-type" content="text/html; charset=ISO-8859
-1"><title>Google</title><style>body,td,a,p,.h{font-family:arial,sans-serif}.h{f
ont-size:20px}.h{color:#3366cc}.q{color:#00cc}.ts td{padding:0}.ts{border-collaps
e:collapse}#gbar{height:22px;padding-left:2px}.gbh,.gbd{border-top:1px solid #c9
d7f1;font-size:1px}.gbh{height:0;position:absolute;top:24px;width:100%}#ghi,.ghs
{background:#fff;left:0;position:absolute;top:24px;visibility:hidden;z-index:100
0}#ghi{border:1px solid;border-color:#c9d7f1 #36c #36c #a2bae7;z-index:1001}#gus
er{padding-bottom:7px !important}#gbar,#guser{font-size:13px;padding-top:1px !im
portant}@media all{.gb1,.gb3{height:22px;margin-right:.73em;vertical-align:top}#
gbar{float:left}.gb2{display:block;padding:.2em .5em}a.gb1,a.gb2,a.gb3{color:#0
0c !important}.gb2,.gb3{text-decoration:none}a.gb2:hover{background:#36c;color:#
fff !important}</style><script>>window.google=<kEI:"aF7HSLvYEKXYigHQqMTZBQ",kEXPI
...</script>
```

The interface includes a menu bar (Display, Port, Capture, Pins, Send, Echo Port, I2C, I2C-2, I2CMisc, Misc), buttons for '\n', Clear, and Freeze, and a configuration panel with settings for Baud (9600), Port (3), Parity (None), Data Bits (8 bits), Stop Bits (1 bit), Hardware Flow Control (None), and Software Flow Control. A status panel on the right shows connection status and various control signals. The bottom status bar displays 'UART receiver framing error', 'Char Count:12560', 'CPS:0', and 'Port: 3 9600 8N1 None'.

Again, if everything goes well, you will see the HTTP response header followed by the HTML content of the page.

How do I get my Web pages onto the MatchPort AR?

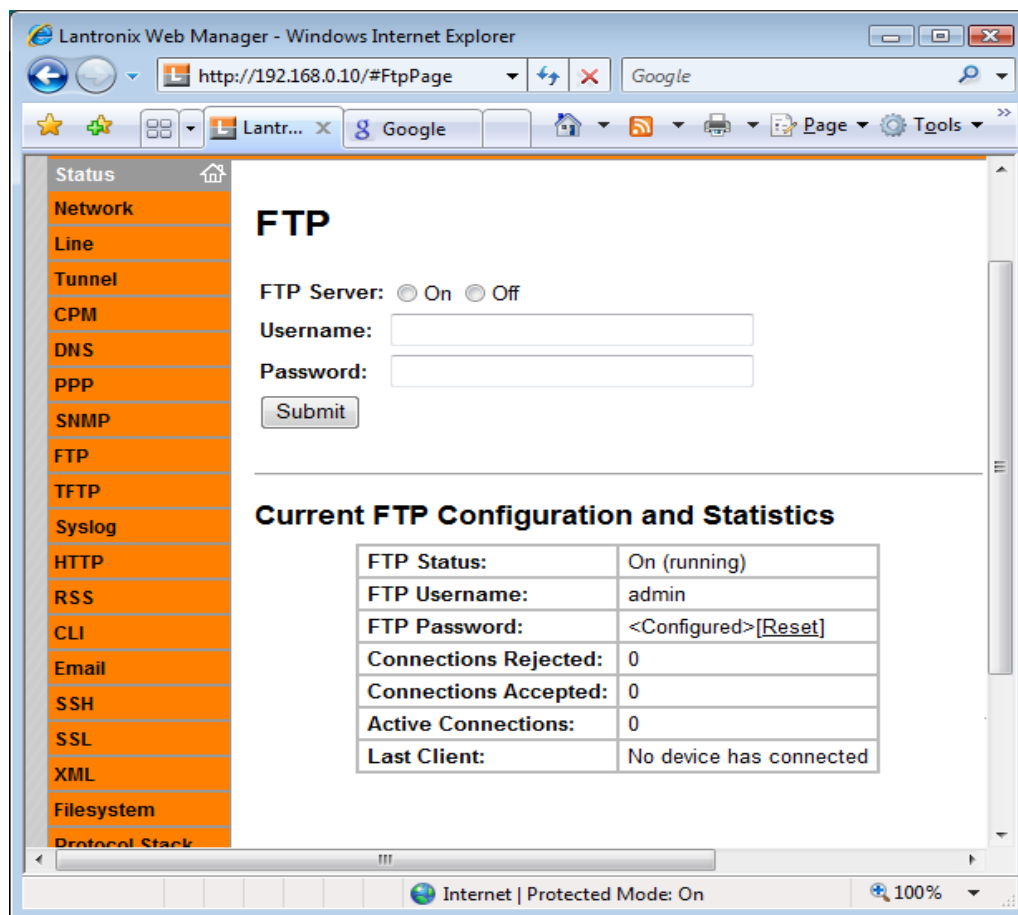
The MatchPort AR comes with software for loading the firmware, including web pages. But if you aren't going to upgrade the firmware version, the easiest way to put new web pages on the MAR is either through the **File System** configuration page in the Web configuration application or with an FTP client.

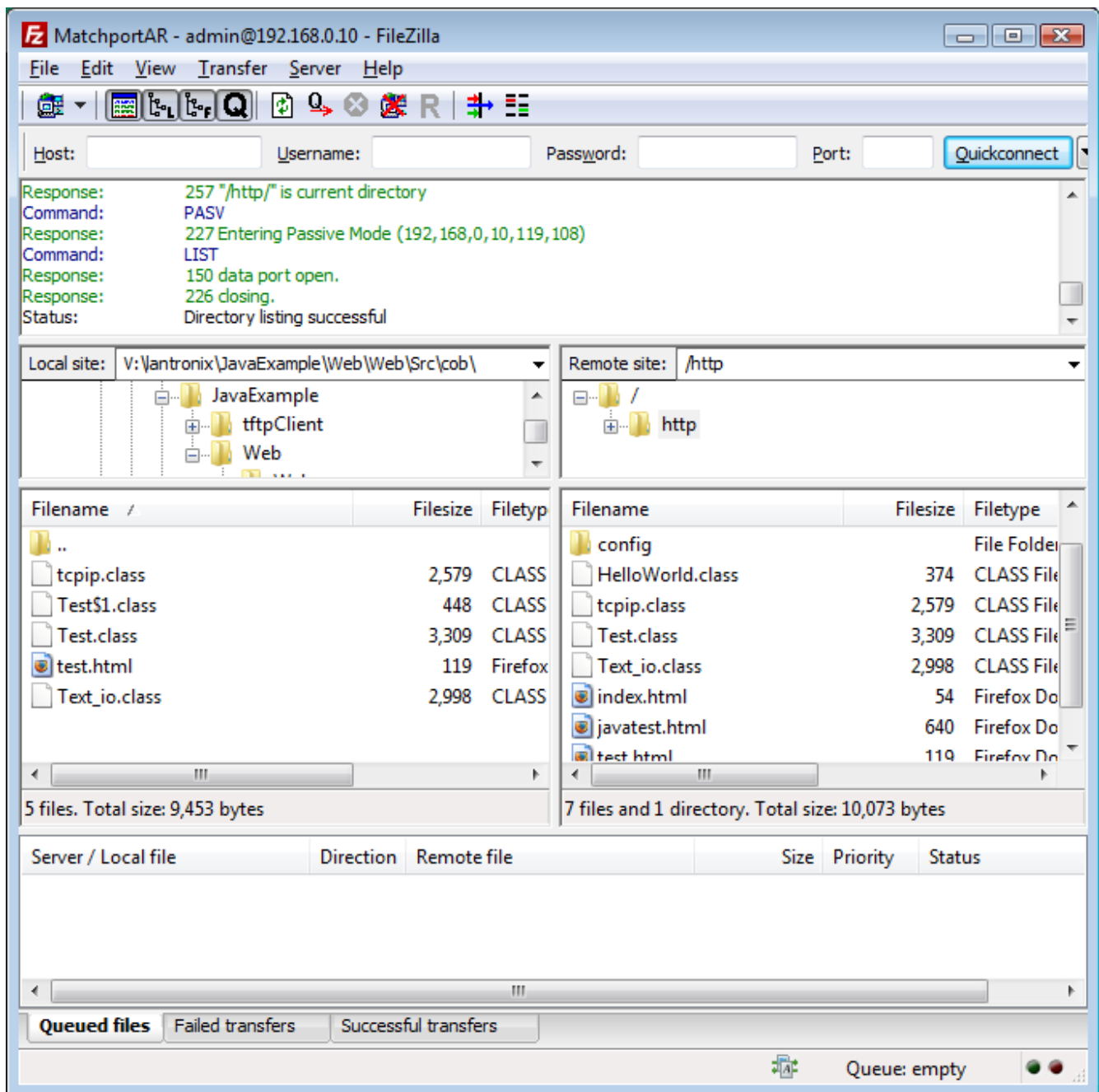
Example: FTP new pages to the MatchPort AR

Configure the FTP server. I think this is set to “On” by default and the login credentials are the same as the default administration login credentials. Now, it is straightforward to use an FTP client such as Filezilla (<http://filezilla-project.org/>) to upload new files to the MAR.

In the second screen shot, I have just uploaded `Test.html`, `Text_io.class`, `Test.class` and `tcpip.class` to the `http` directory of the MAR web server. (More information regarding an example using these files is given below). The configuration application lives in the `config` directory. Don't overwrite any of the files in this directory or you will need to fix them before you can access the configuration application via the Web.

In fact, now is a good time to back them up by using the Filezilla session to copy the `config` files to your computer hard drive. If you want to see how the configuration application works, you can browse these files with a text editor.





How do Web pages access the serial ports?

They don't. This was my big misperception of the development paradigm of the MatchPort AR. I assumed there was a way to load a register or write a file or edit an HTML page from the serial port. For example, imagine that I want to monitor the temperature of some remote network location. I use a micro controller with a temperature sensor to collect temperature data, then write the data *to the MAR* through the serial port.

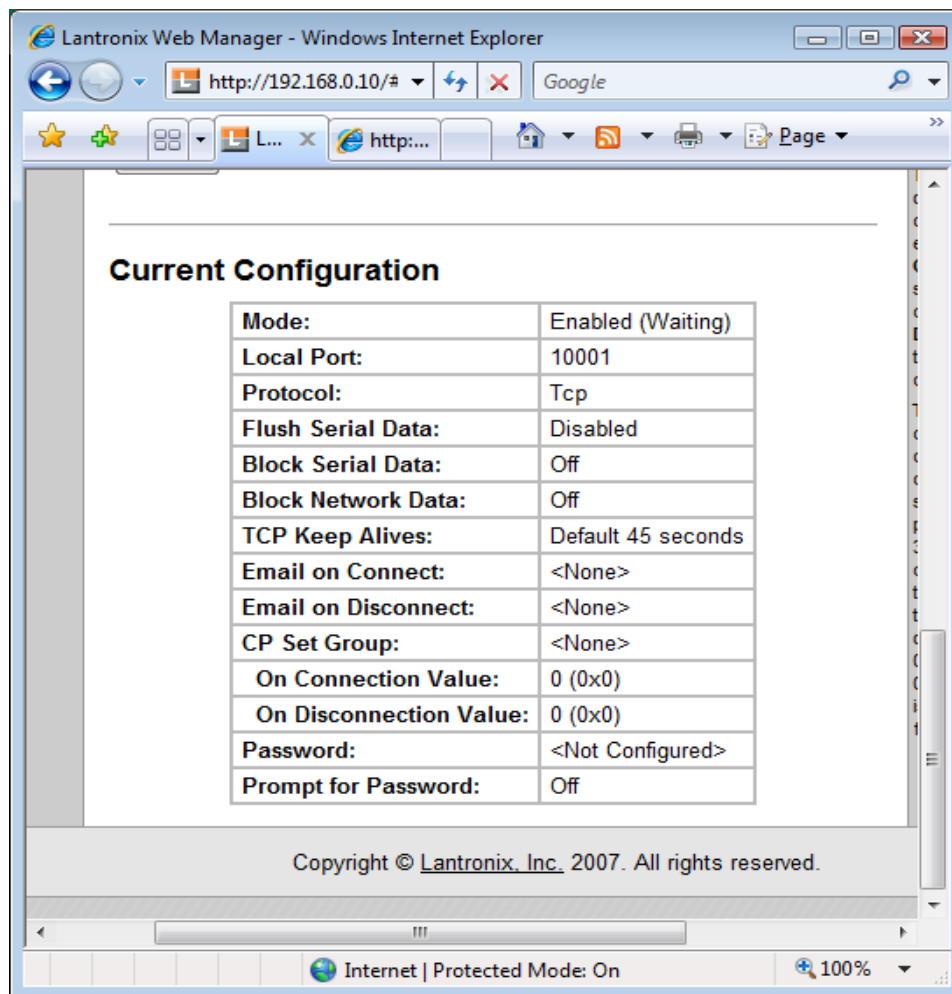
But this isn't how the MAR was meant to be used. When you want to access the serial port from a Web

page, you write a Java Applet that runs on the Java-enabled browser of the client machine and accesses the serial port through a tunnel (e.g. a TCP or SSH tunnel). The MAR serves the Web pages, but is transparent to the serial data, merely passing along the TCP connection to the client Java Applet.

Lantronix has a working example of a Java Applet (fairly well hidden) in their support forum. If you want to build a Web application that also grabs data from the Serial Ports, you will want to read the "Support Answer" titled: *Web Enabling Your Serial Device*. It is at the address <http://tinyurl.com/642l6q>. Download the example Java Applet at the bottom of the page or at <http://tinyurl.com/62mftn>. The files in this package can be loaded into the MAR with your favorite FTP client as in the previous example.

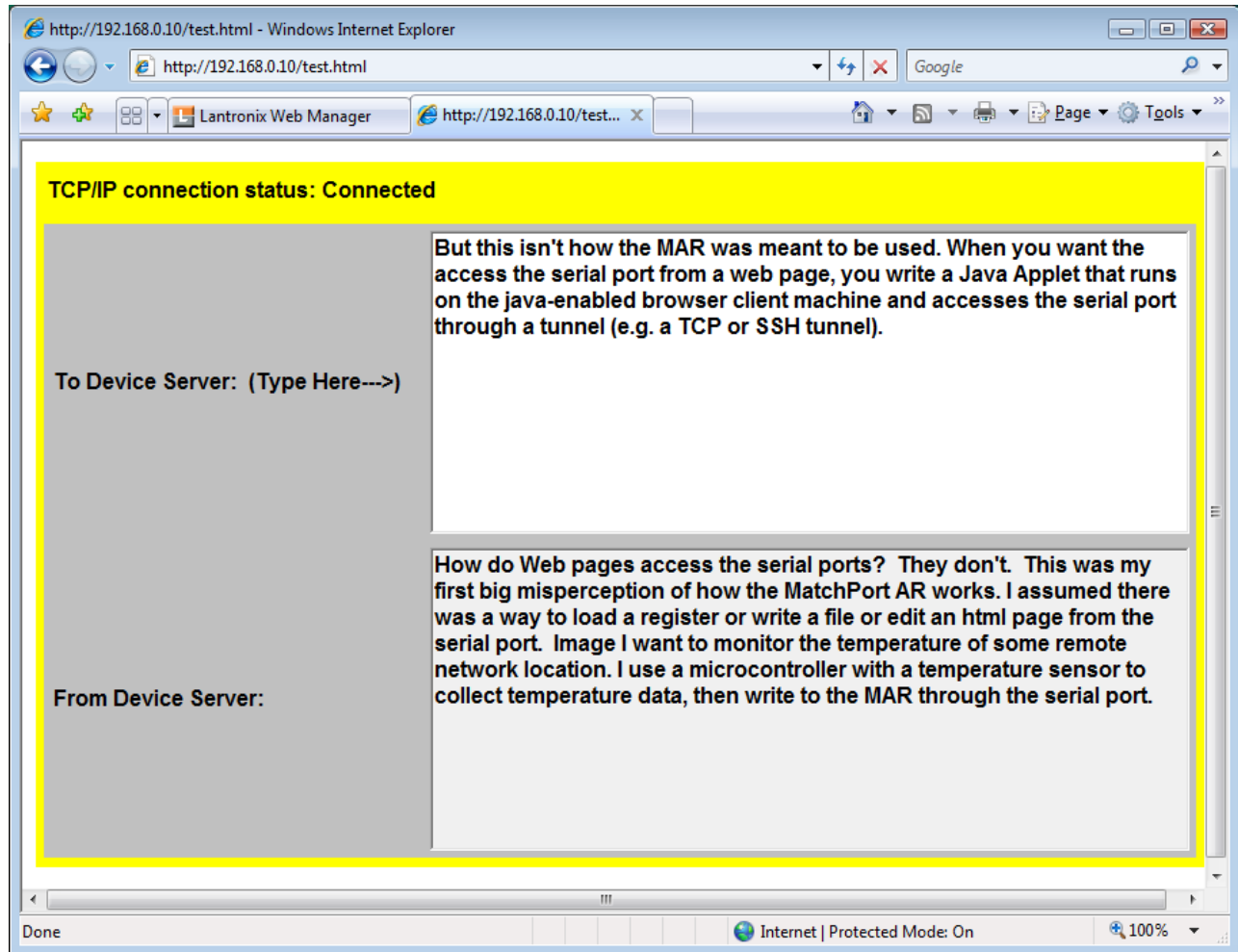
Example: Text transferred via TCP tunnel between Serial Line 1 and Java Applet

When the test page is accessed from a browser on the MatchPort AR, the Java applet we transferred to the MAR in the example above is run in the browser. The applet opens a TCP socket to the MAR. I configured the MAR in Accept Mode listening on port 10001. I disabled Connect Mode, but I don't think that is necessary.

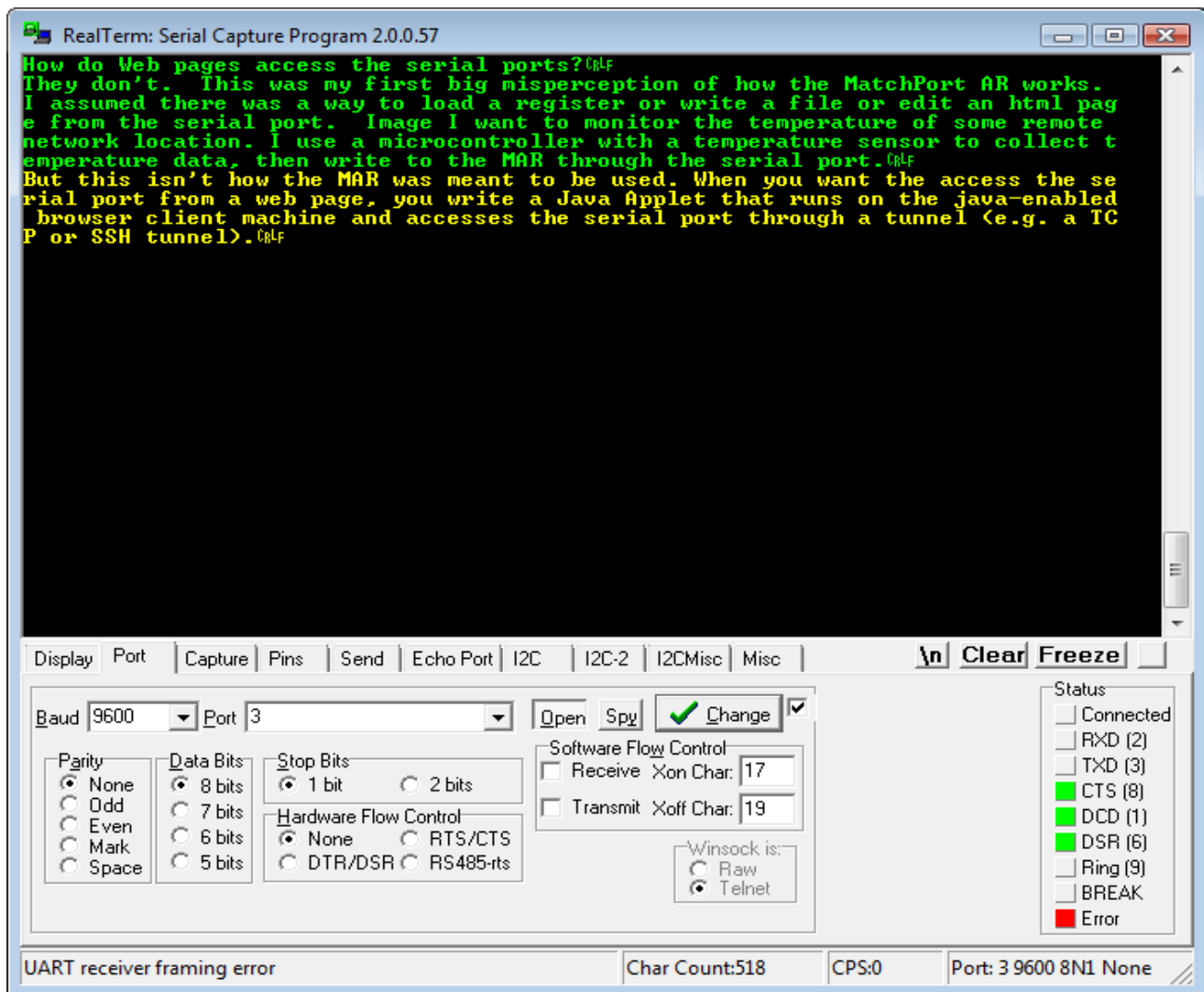


When you type in the “To Device Server” screen, the text appears in yellow in the RealTerm terminal; when you type in the RealTerm terminal, the text, echoed in green, appears in the Java Applet's “From Device Server” window. See screen shots below.

Java Applet Running in Browser—Tunnel to Serial Line 1



RealTerm Serial Line 1—Tunnel To Java Applet



Configuring SSH Keys

Configuring SSH keys is similar to any Unix system, so there are many how-to pages on the web. I found that PuTTY (no SSH in RealTerm) generates a key file in a format the MAR doesn't read. It was useful to first have the MAR generate a key pair for its SSH Server configuration (using the Web interface, for example) and then make sure that my PuTTY keys were uploaded from a file in exactly the same format as those generated and downloaded from the MAR.

I hope you found this useful. If you have question or comments, contact me at scott@drskippy.net.



MatchPort AR Missing Tutorial by [Scott Hendrickson](#) is licensed under a [Creative Commons Attribution-Share Alike 3.0 United States License](#) (<http://creativecommons.org/licenses/by-sa/3.0/us/>).

Based on a work at blog.drskippy.net.

Permissions beyond the scope of this license may be available at <http://drskippy.net>.