Apr 11, 2002

APPLICATION NOTE 198

Networked Temperature Monitoring

An IP-based networked sensor monitor can easily be created through the combination of the Tiny Internet Interfaces (TINI®) platform, 1-Wire® sensors, and the appropriate Java™ software. The TINI platform provides the TCP/IP network stack and the local control capabilities required to design IP-based, networked sensors. The included Java runtime environment and the 1-Wire peripheral interface library, allow for easy control and communications using 1-Wire devices.

This application note demonstrates an IP-based networked temperature monitor, with a downloadable applet control interface that executes in Java enabled browsers. It utilizes a TINI Verification Module and a DS1920 iButton® or DS1820 1-Wire temperature sensor. The applet controls the sensor and displays the time and temperature samples taken. The applet is automatically downloaded by browsing to the IP address of the TINI, and is served using the TINI Runtime Environment.

Introduction

The Tiny InterNet Interfaces (TINI®) platform provides the TCP/IP network stack and the local control capabilities required to design IP-based, networked sensors. The included Java™ runtime environment and the 1-Wire® peripheral interface library, allow for easy control and communications using 1-Wire devices. This application note demonstrates an IP-based networked temperature monitor, with a downloadable applet control interface that executes in Java enabled browsers. It utilizes a TINI Verification Module and a DS1920 iButton® or DS1820 1-Wire temperature sensor. The applet controls the sensor and displays the time and temperature samples taken. The applet is automatically downloaded by browsing to the IP address of the TINI, and is served using the TINI Runtime Environment.

System Overview

This application utilizes either a TINIm390 or TINIm400 Verification Module. The DS1920 is connected to the TINI's External 1-Wire bus. The TINI provides Ethernet connectivity and functions as the 1-Wire master.

TINI 1-Wire Interface Library

The TINI Runtime Environment's API supports 1-Wire devices with libraries for adaptors and 1-Wire specific containers exposing their individual functionality. This application uses DSPortAdapter, and OneWireContainer10, which contain methods needed to communicate with the DS1920. To begin 1-Wire communications, the TINI requests the default adapter:

```
DSPortAdapter ourAccess = OneWireAccessProvider.getDefaultAdapter();
```

Next, the TINI targets the desired family code for the 1-Wire device (0x10 in this case), and receives a reference to its container.

```
ourAccess.targetFamily(0x10);
ourAccess.findFirstDevice();
```

```
OneWireContainer10 tc = ( OneWireContainer10 ) ourAccess.getDeviceContainer();
```

With the referenced container, the TINI now has access to all features of the DS1920. Features of the iButton are exposed to the application using the methods below.

```
readDevice();
writeDevice();
doTemperatureConvert(byte[] state);
getTemperature(byte[] state);
setTemperatureAlarm(int alarmType, double alarmValue, byte[] state);
isAlarming(); (inherited from OneWireContainer)
```

To sense the temperature, read the state information of the iButton with readDevice, then use this to perform a temperature measurement by calling doTemperatureConvert followed by getTemperature to read the converted temperature.

The DS1920 has built-in high and low temperature alarms that can be set programmatically to notify the master when a measured temperature value is outside the set range. To use the temperature alarms, set them with the setTemperatureAlarm method passing TemperatureContainer.ALARM_HIGH or TemperatureContainer.ALARM_LOW as parameters, as well as the new alarm value and the state. Calling the isAlarming method on the container allows you to monitor the state of the set alarms.

In this application, ButtonControl.java handles the temperature samples while AlarmMonitor.java checks for alarms.

TINI Server Software

The TINI implements a simple web server for incoming HTTP requests. Figure 1 shows the interactions between the client and server. It is broken into four threads showing the individual parts of the TINI temperature server. It services connections with a simple HTML index page containing an embedded applet. This applet is downloaded by the client and used to control the temperature sampling and display the data. The web server is implemented by TempButtonHost.java.

TINI also maintains a server socket waiting for inbound connections. When a connection is made, SockListen.java deciphers the command and calls TempCommand.java to execute it. TempCommand calls the required methods to carry out the command and utilizes AppletComm.java to send information back to the client if necessary.

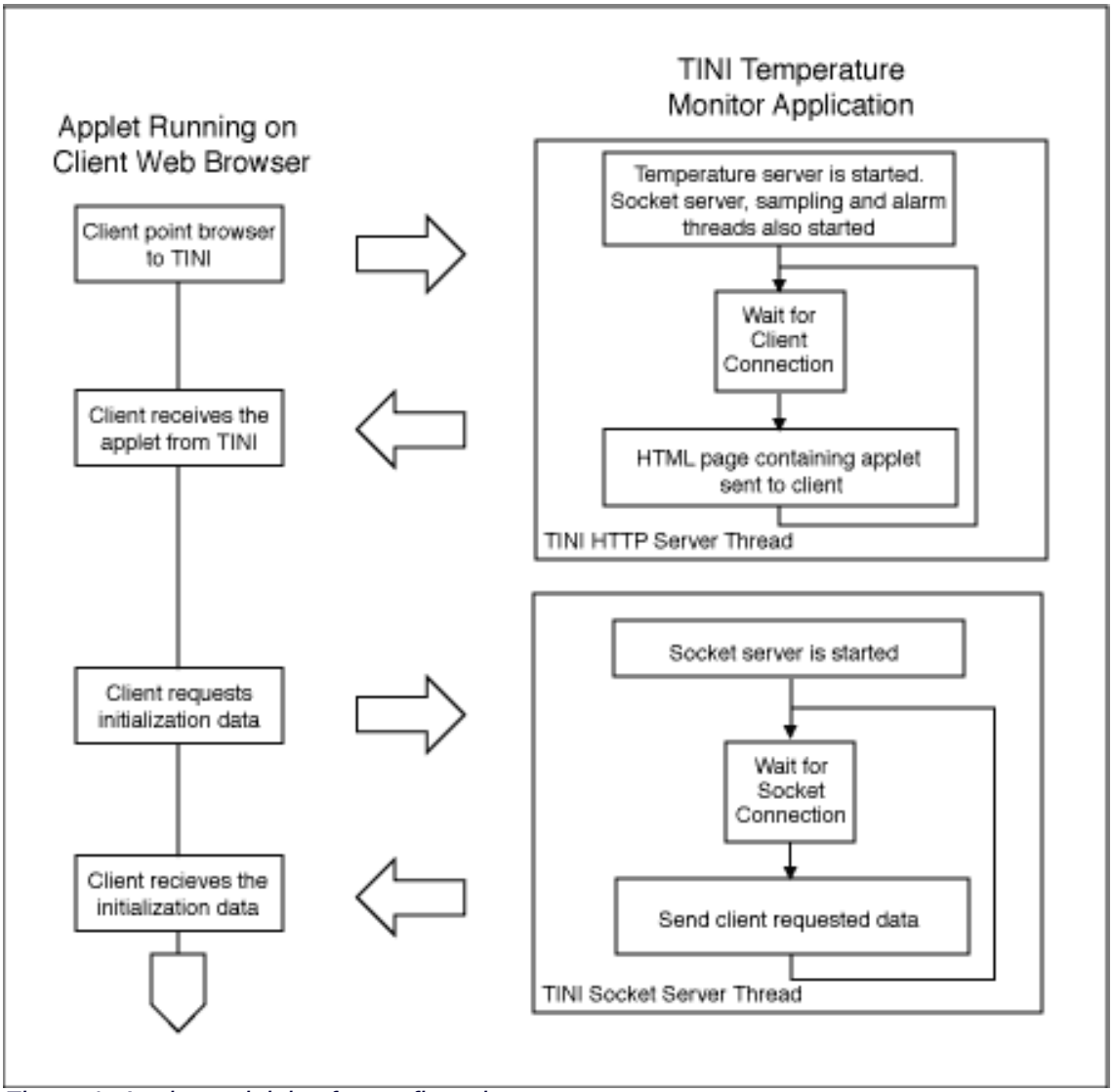

Figure 1. Applet and tini software flow chart.

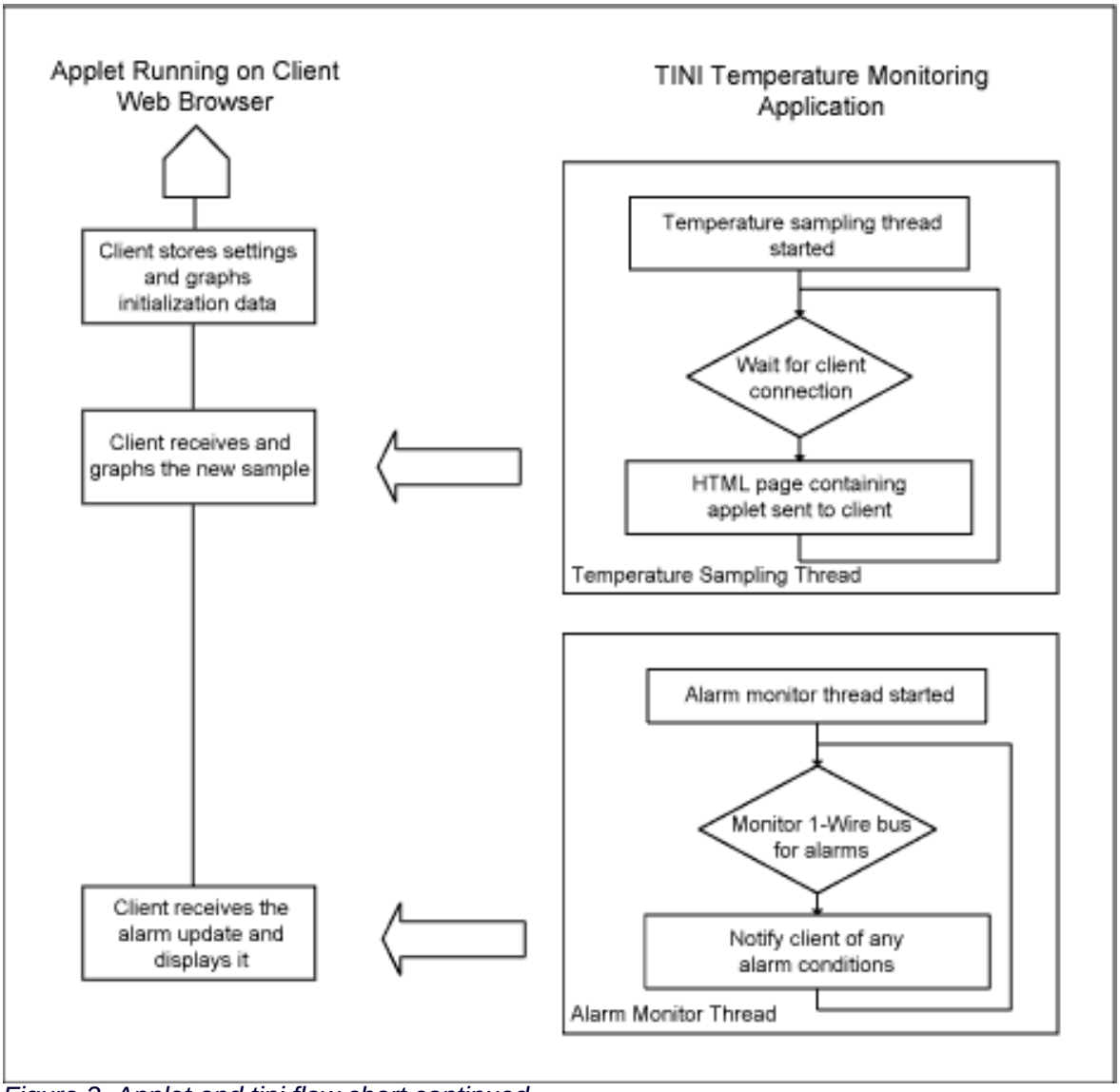

Figure 2. Applet and tini flow chart continued.

TINI Sample Buffer

The TINI, using SampleHold.java maintains a time and temperature buffer, which stores up to 300 samples in TINI's file system so the sample data is persistent. SampleHold uses a RandomAccessFile to hold all the information required retrive the time and temperature sampling application.

Temperature iButton Control Applet

The control applet displayed is Figure 3. The primary purpose of the applet is to control the sampling of the DS1920, and to display the samples in a meaningful format. The graphs outline changes between red, blue, and black depending on the current alarm condition. The current temperature of the TINI is displayed below the graph; both Fahrenheit and Celsius scales are available. The graph is scalable along the temperature and time axes to make the sampled data easier to interpret. Both the high and low trip points are shown on the graph when the range is correct, the high in red and the low in blue. These are adjusted from the controls to the right of the graph. The applet also has control over the number of samples to take during a given period of time. It also provides controls to start and stop the sampling, clear the buffer and to shut down the temperature server. If a shutdown command is received, all applet configuration data is saved so the applet can maintain state across sessions.

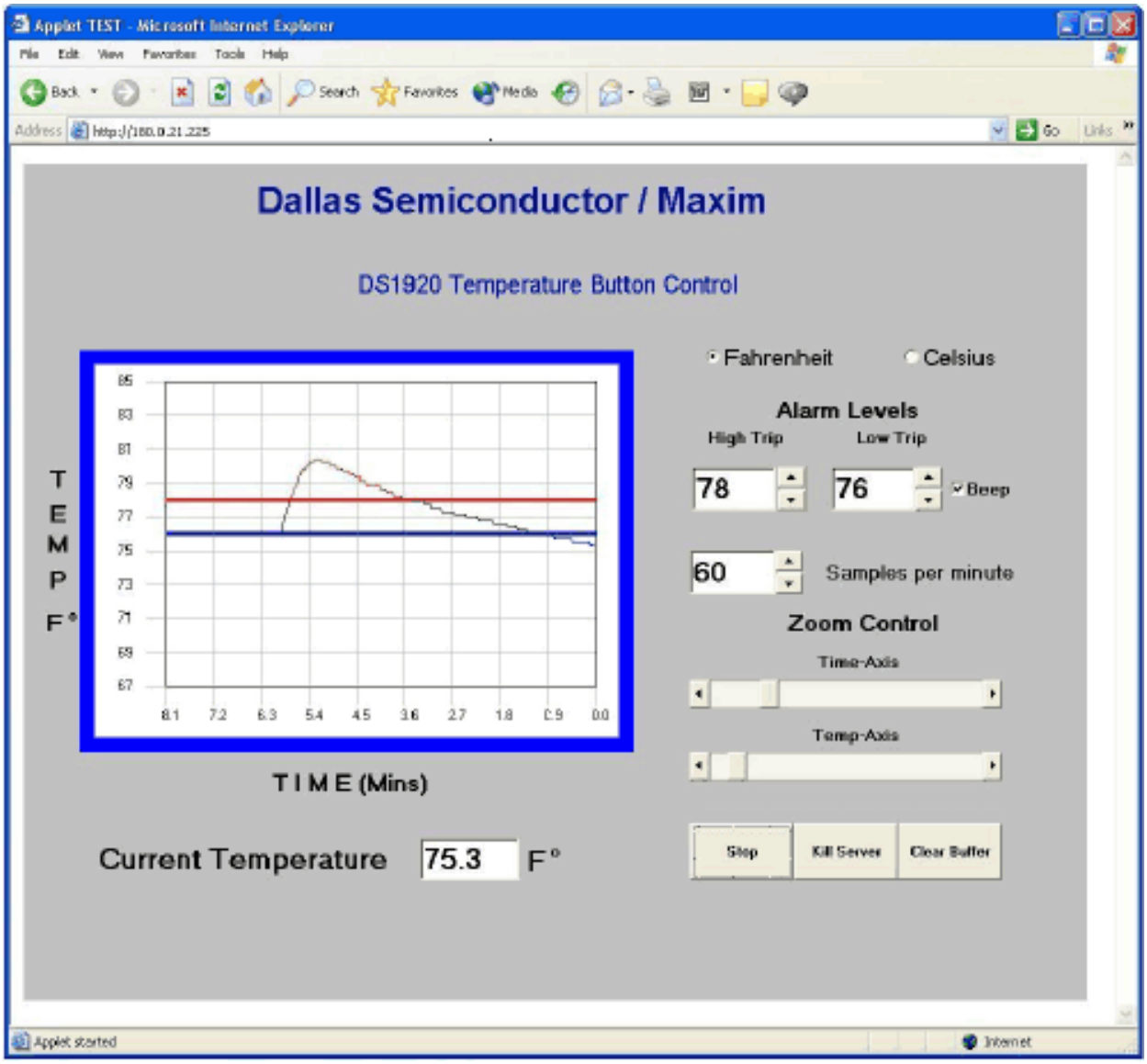

Figure 3. Temperature control applet.

The Applet consists of several classes:

- TempApplet.java is the main class implementing AWT content.
- TempGraph.java implements the graph of temperature versus time.
- TiniComm.java, TiniListen.java communicates between the TINI and the applet.

Associated Files

A file associated with AN198 is located at ftp://ftp.dalsemi.com/pub/tini/appnotes/AN198/AN198.zip.

Conclusion

Using the TINI Runtime Environment and the 1-Wire library, a sophisticated temperature-sampling device can easily be created. This application shows how the TINI can best be used as a remote interface device for taking, and storing samples and serving this data to a client to handle the computationally intense task of displaying the data.

Dallas Semiconductor/Maxim Contact Information
Company Addresses:

Maxim Integrated Products, Inc.
120 San Gabriel Drive
Sunnyvale, CA 94086
Tel: 408-737-7600
Fax: 408-737-7194

Dallas Semiconductor
4401 S. Beltwood Parkway
Dallas, TX 75244
Tel: 972-371-4448
Fax: 972-371-4799

Product Literature/Samples Requests:
800-998-8800
408-737-7600

World Wide Website:
www.maxim-ic.com

Product Information:
http://www.maxim-ic.com/MaximProducts/products.htm

Ordering Information:
http://www.maxim-ic.com/BuyMaxim/Sales.htm

FTP Site:
ftp://ftp.dalsemi.com

More Information

DS1820: QuickView -- Full (PDF) Data Sheet

DS1920: QuickView -- Full (PDF) Data Sheet -- Free Samples