



Finger Lakes Engineering
119 South Cayuga Street
Suite #200
Ithaca, Ny 14850

PHONE:
(607)-277-1614

FAX:
(800)-835-7164

FLE's President:
steve@flconsult.com

WEB:
WWW.FL-ENG.COM

Introduction

In December 2005, I purchased a dual-temperature wine-cooler for my wife's Red and White wines. We aren't wine connoisseurs by any stretch of the phrase and here in the Finger Lakes region, there are plenty of great wines in \$8 to \$15 cost point.

In looking for a practical application note to cover all system-level items that one would encounter when developing an Ethernet appliance, the wine-cooler became a great choice for an application note on the Mustang.

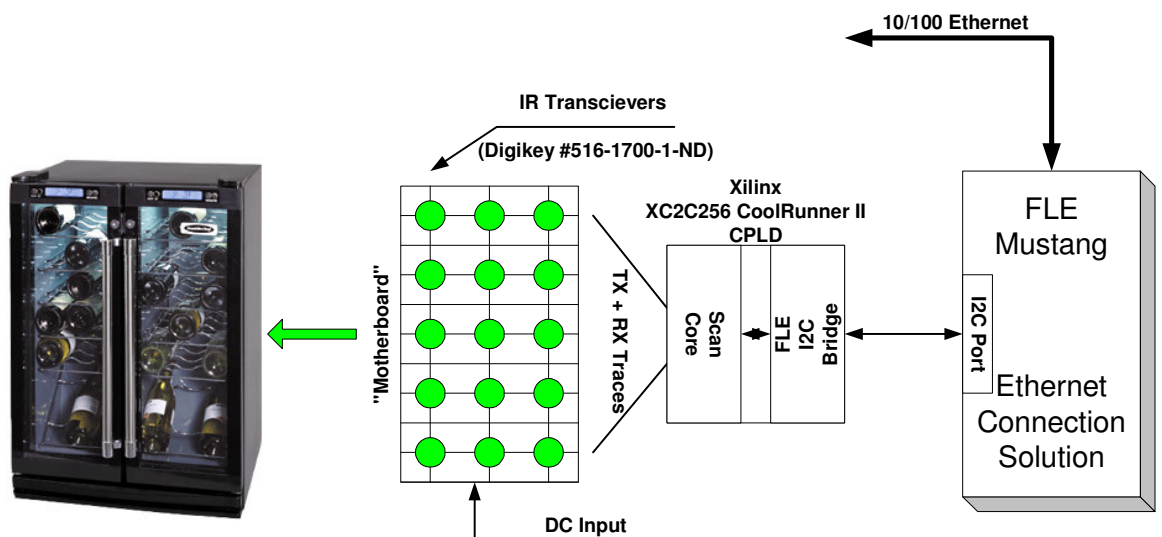
In this application note, we will show how the FLE Mustang's hardware architecture, firmware, and PC connectivity allow the development of a fully-automated wine-cooler. The cooler can keep track of inventory, notify me via email when it is running low, and even order more wine for us when it gets low!

Design Requirements

The design requirements for the wine-cooler include the ability to monitor the stock state of 36 bottles of wine (in-stock/out-of-stock), provide 10/100 wired Ethernet connectivity, allow the user to login and check the wine cooler via a Java GUI, and enable email alerts when the wine supply drops below a user defined level. The power source was specified to be an AC/DC adapter providing 3.3V at 200mA.

Hardware Architecture

The first task in developing the wine-cooler includes the ability to monitor the stock state of 36 wine bottles. Using the Mustang's simple-to-interface motherboard expansion module, a Xilinx CPLD using an I2C interface, and an IR transmitter/receiver matrix; we can leverage the power of the Mustang to complete our hardware design. The block diagram of the system is shown below.



A simple 2-layer "Motherboard" was developed that employed a matrix of 36 individual IRDA infra-red transceivers. Each IRDA device was independently wired to a Xilinx XC2C256 CoolRunner II CPLD. The CoolRunner consisted of a core that would activate each IRDA sensor, transmit a data pattern, and look for the pattern to be received. If the wine is in stock, the bottle's glass reflects the infrared data back to the receiver and a bit is set (or cleared) in a register within the CPLD.

The FLE Mustang's custom user code queries the CPLD scanner core using the I2C port on the Mustang's Motherboard header. This simple data connection allows almost infinite expansion capabilities of the I/O ports on the Mustang. In this case, the Mustang used the equivalent of 72 GPIO ports (36 transmit and 36 receive) by required only 2 pins on the Mustang PCB.

The motherboard carried the IRDA sensors, CPLD, Mustang, and a DC barrel connector for an external AC/DC power supply port. The Mustang OEM module provided the RJ45 connection for my home's broadband network.

This example shows how some simple hardware engineering, paired with the Mustang, can easily provide a product-ready hardware implementation without being I/O limited.

Firmware Architecture

Now that a hardware design was in place, the firmware of the system needs to be developed. The firmware will run in the User Application section of the Mustang's code-space and provide the ability to send auto email alerts when the wine cooler is getting low.

Since the Mustang provides FLE's supported LwIP Ethernet stack, I2C Drivers, UART Drivers, and an execution thread; very little code is needed. The pseudo-code for this application looks as follows

```

WineCoolerAlerts()
{
    WaitFor(10 seconds); // we don't drink wine that fast!
    for (n=0;n<7;n++) shelf[n]=ReadI2C(Port0,n) // let's read the in-stock/out-of stock bits for each shelf in the cooler
    for (n=0;n<7;n++)
    {
        if (shelf[n]!=0xFF) // uh oh! We are getting low on some wine!
        {
            WineBit = {code to extract the bits indicating what wine is out-of-stock}
            LabelNameString=ReadEEPROM(WineBit); // using that bit, lets get the name of the bottle from E2
            NotificationName=ReadEEPROM(NotName); // get the email address we need to alert

            ... Build a Cute Message to tell me what Wine Bottle is Out Of Stock ...

            LwIPSendEmail(NotificationName, Subject, CutesyMessage); // send me the alert via LwIP SMTP
            // email notification
        }
    }
}

```

This WineCoolerAlerts routine is all that is needed within the Mustang FLE provided code-set to enable email alerts from the wine cooler. A simple routine reads the I2C port to determine the stock status, compiles a little message, and issues an SMTP based email via the LwIP Ethernet stack provided by FLE.

The pre-existing functionality of the Mustang also allows for web-page storage and retrieval. Using a Java GUI and the FLE Mustang Loader application, we will place a GUI on the Mustang board so we can get a UI for the wine cooler and load the Label Names, Notification Names, and other variables that the WineCoolerAlerts function needs to be complete.

PC/Web Architecture

In order to really access all the features of an embedded appliance, an executable application is the best choice. An executable provides TCP/IP socket access, file save/load features, graphic capabilities, error checking, and many other features taken for granted on every PC application in the world.

For the wine cooler, a simple Java application was developed as the executable. A Java application can be stored as a JAR file (essentially a zip) on the Mustang board along with an index.html to point to the Java application. Logging into the Mustang, via a web-browser, will open the Java application directly within the user's web-browser. From within the Java application, it's possible to open/close TCP/IP sockets and this provides one method of communicating with the Mustang.

Upon startup, the Java application opens a TCP/IP socket to the Mustang to query the I2C port and the EEPROM. The data is downloaded into the GUI to show the names and stock status of each wine bottle. From this application, the user can then change the names of the bottles, set email alerts, and configure the wine cooler's options. The Java method provides the power of a Windows/PC executable application to be used to interface with the wine cooler. Java allows for future applications to build upon a common platform of libraries without being limited to simple HTML functions.

Summary

This application note explored a fun experiment with the FLE Mustang to create an Ethernet based wine-cooler that provides on-demand stock status and email-notifications. This note showed how the Hardware, Firmware, and PC/Web architecture can easily be created using the Mustang to implement the design goals. In a future application, it could be possible to use the encryption engine of the Mustang to store a credit card number and automatically order wine directly from the vendors. This started as a fun application, however, an auto-ordering wine cooler might be a great product anyway!.

More application notes on the Mustang, focusing on elements of this example will be available soon on www.FL-ENG.com