WHITE PAPER

TEXAS INSTRUMENTS

Jean Anne Booth
Director of WW Stellaris Marketing
and Customer-Facing Engineering
Sue Cozart
Applications Engineer

Access Your Embedded Controller with Ease through a Web Server

Introduction

Besides running the primary control application for an embedded system, a single Stellaris MCU can also manage the HMI as a dynamic Web server so the system can be readily monitored and controlled – from across the country, if needed.

The panel through which an operator interacts with a piece of equipment is an important and frequently-used feature of that equipment. Information about the equipment—its settings, operational status, environmental conditions, and production output—can help the user follow progress, take corrective action, make adjustments, or optimize the equipment's use. The information might also be gathered for off-line long-term analysis of the system. More information is better than less and the more easily it is obtained, the better.

The designer should also make it as easy as possible to change the settings on a piece of equipment, reconfigure its operation, or fine-tune the system. The more intuitive and explicit that activity is, the more likely the result will be what the operator desires. Losing the instruction manual can seriously impair the user's operation of many systems.

The Human-Machine Interface

The design of the human-machine interface (HMI) of a piece of equipment can be key to its effective use. The HMI can be a physical part of the system, like an LCD panel for the display with buttons and knobs for various control functions. Alternatively, it can be a virtual interface, implemented in software and run from a common terminal of the user's choosing. That terminal could be a PC, laptop, cell phone, or purpose-made handheld device. With a virtual HMI, especially when accessible over a network, the operator can be standing at the machine or at a console in a different, environmentally-comfortable room—or even across the country—and still manage, monitor, and control the system.

A physical HMI will usually reflect the sophistication of the interface. A small panel with a few buttons will only be able to deal with a small amount of information and offer limited control. It takes a larger display and more complex inputs to support more intricate management. Touch screens are a popular means of eliminating troublesome buttons, but are also limited by the capabilities and characteristics of the underlying screen.

With a virtual HMI there is no appreciable impact on the system cost or size, even with a sophisticated information and control center. One basic design can span numerous product categories, serve multiple levels of system management, and be expanded to add new features—even in the field.

The Web browser has become a very familiar interface over the last 20 years, has evolved to accommodate many forms of media, and is found in all shapes and sizes on all manner of devices. The same browser interface might see a child perform educational activities at her favorite Web site one moment, let a consumer securely verify a bank transaction, and enable a petroleum engineer in Lafayette, Louisiana to check the pressure in a line on an oil rig in the Gulf of Mexico and adjust the valve settings.

The ubiquity of the Web browser and the number of browser-enabled terminals make a Web server an ideal vehicle for providing information about a piece of equipment and taking inputs for the control of that equipment. Web servers can be expanded to accommodate advanced features on more sophisticated systems, or be quite simple to provide very basic information and take straightforward inputs.

Because the Web interface is well established and so popular, Web servers are readily available and, running on the right processors, can provide an ideal HMI for embedded controllers in a compact form, using minimal development effort, and requiring negligible overhead. The added advantage of remote monitoring and control made possible over a network connection greatly enhances the applicability of the equipment.

Web Server

Web server applications are designed to respond to requests from the network addressed to the equipment location. When a client device asks for it, the Web server creates the appropriate Web page structure, gathers and inserts relevant data, inserts links to information available outside of the immediate system, readies media for transmission, and encodes the assembled page so it can be transmitted. The server then triggers a series of events that are handled by the network protocol stack which transmits the page to the requesting client over the physical network.

Developing a Web server can seem to be a daunting task, especially for designers and programmers whose real expertise is in motion control, medical technology, fluid mechanics, or audio processing. Fortunately, Web server applications and Transmission Control Protocol/Internet Protocol (TCP/IP) stacks are readily available, sometimes free, and open-source so they can be modified as desired. These software packages are designed to be used by people with minimal understanding of the underlying details so network or Web page proficiency is not requisite.

Two popular network stacks are µIP (micro Internet Protocol) and lwIP (lightweight Internet Protocol) offered by Adam Dunkels (http://www.sics.se/~adam/software.html) These TCP/IP stacks are designed to consume very little code space and use the minimal RAM typically found in MCUs (see Table 1).

Table 1. TCP/IP Stack Comparison

Feature	µIP	lwIP
Stack code space (kB)	4.5	15
RAM requirements (kB)	2.5	20
IP and TCP checksums	x	x

Table 1. TCP/IP Stack Comparison (Continued)

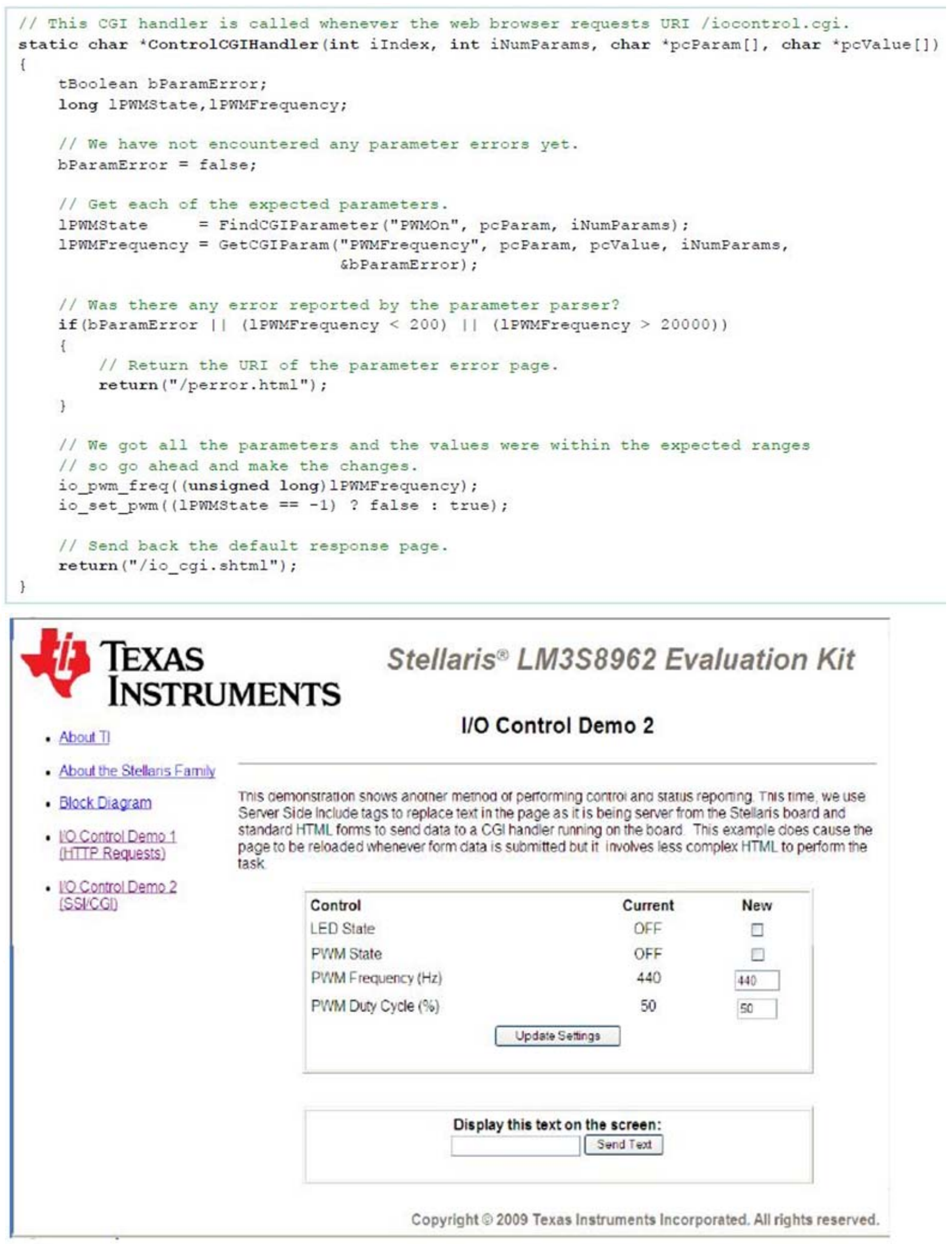
Feature	µIP	lwIP
IP fragment reassembly	x	x
IP options		
Multiple interfaces		x
UDP		x
Multiple TCP connections	x	x
TCP options	x	x
Variable TCP MSS	x	x
RTT estimation	x	x
TCP flow control	x	x
Sliding TCP window		x
TCP congestion control	Not needed	x
Out-of-sequence TCP data		x
TCP urgent data	x	x
Data buffered for rexmit		x

The primary difference between µIP and lwIP are memory requirements and capabilities. The smaller µIP package implements the IP, ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol), and TCP protocols. The lwIP package is larger adding in PPP (Point-to-Point Protocol) and DHCP (Dynamic Host Configuration Protocol) plus an operating system emulation layer, buffer and memory management subsystems, and network interface functions as well. Both packages provide support for dynamic Web page delivery.

Both include embedded Web server applications that run on top of these stacks to accept commands for the equipment and to prepare dynamic pages containing status, data, and information from the equipment.

Source of Content

The content of a web page may be static or dynamic. Static pages read the same every time they are accessed. Instructions and help menus may be largely static. More interesting and more useful pages tend to be dynamic as they contain variable information that might show current settings, recent data captures, images, and data streams. Dynamic web pages can be built using a function call to a script or compiled program which uses the common gateway interface (CGI) and server-sided includes (SSI) to merge the most current data into the page as it is assembled at the client from the HTML description. Newer techniques, such as Ajax (asynchronous JavaScript and XML), can be used to create web applications that can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Examples of the code and the resulting Web page are shown in Figure 1.



```
// This CGI handler is called whenever the web browser requests URI /iocontrol.cgi.
static char *ControlCGIHandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    tBoolean bParamError;
    long lPWMState,lPWMFrequency;

    // We have not encountered any parameter errors yet.
    bParamError = false;

    // Get each of the expected parameters.
    lPWMState     = FindCGIParameter("PWMOn", pcParam, iNumParams);
    lPWMFrequency = GetCGIParam("PWMFrequency", pcParam, pcValue, iNumParams,
                                &bParamError);

    // Was there any error reported by the parameter parser?
    if(bParamError || (lPWMFrequency < 200) || (lPWMFrequency > 20000))
    {
        // Return the URI of the parameter error page.
        return("/perror.html");
    }

    // We got all the parameters and the values were within the expected ranges
    // so go ahead and make the changes.
    io_pwm_freq((unsigned long)lPWMFrequency);
    io_set_pwm((lPWMState == -1) ? false : true);

    // Send back the default response page.
    return("/io_cgi.shtml");
}
```

Stellaris® LM3S8962 Evaluation Kit

I/O Control Demo 2

- About TI
- About the Stellaris Family
- Block Diagram
- I/O Control Demo 1 (HTTP Requests)
- I/O Control Demo 2 (SSI/CGI)

This demonstration shows another method of performing control and status reporting. This time, we use Server Side Include tags to replace text in the page as it is being server from the Stellaris board and standard HTML forms to send data to a CGI handler running on the board. This example does cause the page to be reloaded whenever form data is submitted but it involves less complex HTML to perform the task.

Control	Current	New
LED State	OFF	☐
PWM State	OFF	☐
PWM Frequency (Hz)	440	440
PWM Duty Cycle (%)	50	50

Copyright © 2009 Texas Instruments Incorporated. All rights reserved.

Figure 1. *Current data is inserted into the page utilizing a common gateway interface (CGI)*

HTML also has a powerful "link" capability that lets material from different servers be merged for the final page at the client. This capability allows proper corporate servers to provide streaming data like video or audio and rich content like images and photos directly to the client, leaving the embedded Web server to only trigger the page assembly and provide the latest data updates. Such linking can greatly enhance the viewed page without burdening precious embedded system memory.

Creating and Storing HTML

The system designer can prepare Web pages using something as simple as a common text editor to dedicated Web Creation programs like Adobe Dreamweaver. Some photo editors and many other routine programs can also generate the needed HTML page descriptions. The HTML language is not difficult to read and, with a little research, Web pages can be created or modified by hand using a text editor.

The HTML page descriptions can be put into memory on the embedded system via high-level language (like 'C') data structures, compiled and linked into the on-chip Flash or an SDcard as part of the final code module. Alternatively, a simple file system could be set up in Flash where the HTML is read sequentially as needed.

The embedded Web server offers many advantages as a means of displaying system information and controlling the equipment. The footprint in program memory is small, about 20 KB including the lwIP stack. It may take only a few hundred bytes to store the HTML for each page to be displayed, even though the image in the browser is fairly complex or rich in images. The amount of information that is available through the Web server is nearly limitless and can be changed in many ways. Different data can be made available depending on whether the machine operator, service personnel, or production management is requesting it. Changes can be made, even after the equipment is sold and installed, in case a Luminary Micro logo needed to be changed to Texas Instruments.

Cost-Effective Networked Embedded Control – Stellaris

Only powerful microcontrollers can meticulously control motors and actuators utilizing sophisticated algorithms to maintain precise motion control in spite of adverse conditions and varying loads. Networking these embedded controllers permits coordination between a number of motors and allows remote management of expanded, more flexible systems. Remote monitoring and control of the equipment keeps people away from dangerous areas while keeping an even better eye on operation and wear-and-tear of the equipment, maximizing uptime and utilization. Yet the design of such sophisticated embedded controllers can be easy and the cost of the end system can be kept in check when the right components are selected.

The performance requirement of a microcontroller to perform today's advanced motion control applications is beyond the capabilities of 8-bit and even most 16-bit microcontrollers. Digital signal processors (DSPs), hybrid MCUs, and 32-bit MCUs are often called into service to handle precision motors, complex algorithms, leading-edge sensing, and high-end applications. Some handle signal processing without adequate control or data processing capabilities. Often multiple processors must be engaged to run the complete application which highlights mismatches in development tools and the cost of multiple development teams, while the added components loom large.

A single, very capable microcontroller is the best answer, but adding in networking can additionally tax system performance, development effort, and product cost. Stellaris microcontrollers are designed for serious industrial control applications and have the features and performance to allow the integration of networking and an embedded Web-server without detracting from the primary application. Development kits like the EK-LM3S8962 are available providing the right tools and examples to guide the system designer through product development to bring a new application to market quickly and confidently.

A typical embedded control application is shown in Figure 3. Sophisticated algorithms run a set of motors to control motion of four complex robotic arms as they shift pieces from one moving belt to another. This system has many time-critical inputs and outputs working together over networks. Seven control boards operate the distributed control system. CAN is used for board-to-board communication and the overall process is configured and monitored by an embedded Web server running on a Stellaris EK-LM3S8962 evaluation board.

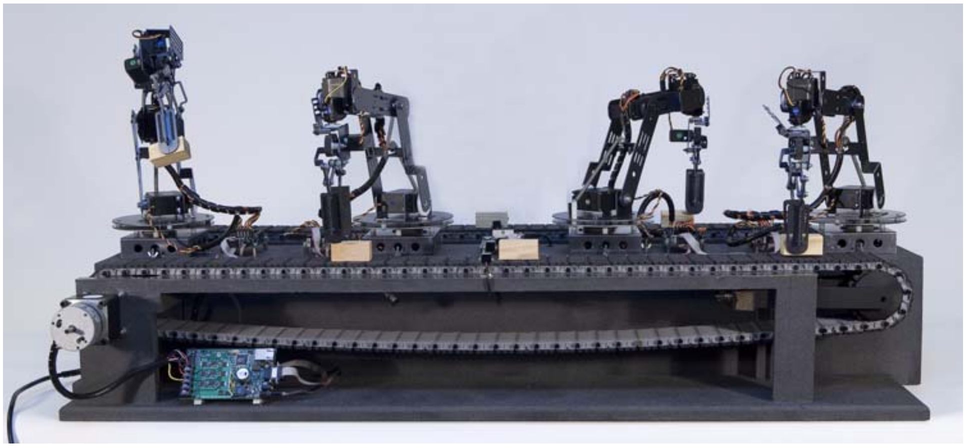


Figure 2. *A Robotic Embedded Control Application Utilizing Ethernet*

While this application is sophisticated motion control, any number of other applications require high performance processing and control along with accessibility via Ethernet. Examples include:

- Numeric Control Machine
- Scientific Instruments
- Instrumentation
- Industrial Control
- Security Systems
- Building Control
- Lighting Control
- Network Switches
- Point Of Sale Terminals
- Medical
- Test & Measurement Equipment
- Valve Controls
- Process Control
- Vision Systems
- HVAC
- Remote Systems Management
- Network Appliances
- Barcode Scanners

Combining as much functionality into a single electronic component as practical has a number of advantages. The higher integration minimizes chip count, board space, interconnects, signal degradation, noise, power consumption, and cost. A few microcontrollers have come on the market the last few years which include an

Ethernet controller but they are still fairly rare and many have capabilities limited by their weak processor.

The Stellaris 6000, 8000, and 9000 series of microcontrollers include not only the Ethernet media access controller (MAC) but the physical interface (PHY) on-chip as well. The MAC (a component of the Data Link Layer - Layer 2 of the network layer model) processes Ethernet frames during transmission and reception. The PHY interfaces to the network and has the ability to scramble and descramble packets. With the MAC and PHY consolidated on the MCU, the system designer knows these functions are matched, operating, and tested as a unit. All that is needed to connect to a network is an RJ45 jack and isolation magnetics plus the Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX (100 Mbps) applications or Cat-3 cable for 10 Mbps networks. The bill of materials (BOM) couldn't get much simpler.

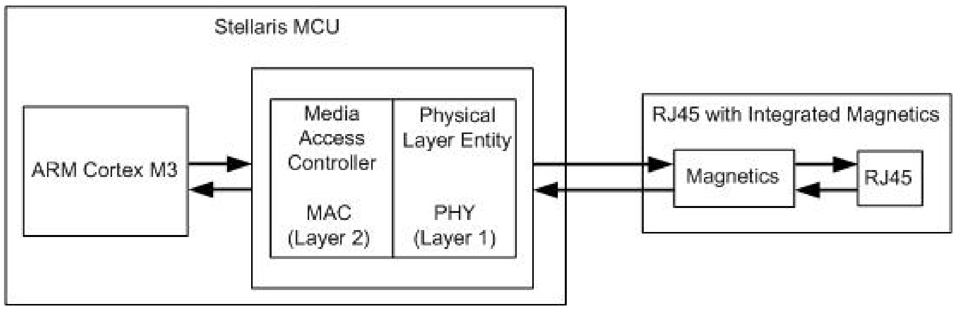


Figure 3. *Many Stellaris Microcontrollers Integrate a 10/100 Ethernet MAC and the PHY On-Chip to Build a nearly Single-Chip Networked Embedded Controller*

There are many other features that make Stellaris microcontrollers an ideal platform for networked embedded control, especially those utilizing Web servers for monitoring and control. Obviously to run a network and support Web pages will require both spare performance and additional memory.

With an ARM Cortex-M3 processor at its heart, a powerful, widely-sourced and broadly-supported 32-bit architecture offers performance to complete the most difficult embedded control tasks. The Cortex-M3 improves execution efficiency and code density by 25% over most 32-bit instruction set architectures to squeeze more performance from the processor while using far less memory. This gives plenty of horsepower to the primary controller application while conserving precious memory.

A wide range of memory combinations are available in Stellaris MCUs from 16 KB of Flash and 2 KB of SRAM to 256 KB of Flash and 96 KB of SRAM. The ratio of RAM to Flash is plenty high to deal with the most complex networking needs, motion control algorithms, data-processing (not just data-logging) routines. Plus the single-cycle Flash accesses keep the processor from bogging down. In addition, many family members have on-chip ROM that frees up the on-chip Flash memory for other user functions by storing the peripheral APIs, Advanced Encryption Standard (AES) tables for cryptography, Cyclic Redundancy Check (CRC) functionality for error detection, and the SafeRTOS™ kernel for general RTOS and safety critical applications.

The peripheral blocks shown in Figure 4 on page 9 show traditional MCU peripherals and important system features plus a solid analog section handling up up to 1 MSPS through 10-bit analog-to-digital converters. Fast interrupt response and a well-coordinated interrupt handler lets all these peripherals dance together without missing a beat, whether they're running motor control algorithms, other primary applications, or multiple networks.

The deterministic architecture of the ARM Cortex-M3 and the low overhead of communicating with the on-chip peripherals allow Stellaris MCUs to offer superior integrated motion control and connectivity in a single chip. The powerful pulse-width modulators (PWM) and quadrature encoders greatly ease motor control operations for precision motion control and offload these strenuous activities from the ARM processor. A single MCU can simultaneously control a variety of motors with up to 8 general-purpose PWMs and up to 8 channels of motion-control PWMs with dead-band generators providing shoot-through protection for applications such as 3-phase inverter bridges. Important safety features are also integrated, such as fault conditioning for each of the four motion-control PWM output pairs in order to provide quick motor shutdown in low latency situations, synchronization of timers enabling precise alignment of all edges, and quadrature encoder inputs for precise positioning sensing.

From an analog standpoint, Stellaris microcontrollers feature both analog and digital comparators to trigger the accurate analog-to-digital converter (ADC) and to trigger an interrupt when needed, which is useful for infrequent out-of-range events such as a current or voltage spike. This capability eliminates the performance-wasting requirement of constant CPU polling. The ADCs and PWMs are closely integrated; either one can directly trigger the other under a variety of conditions without intervention by the CPU. An internal temperature sensor is also provided, which can be used to monitor and shut down an appliance if it overheats. The dual ADC units provide simultaneous analog measurement capability making Stellaris the ideal solution for demanding motion applications that require simultaneous current and voltage measurements or simultaneous multi-phase measurements, including sophisticated motor drives for AC Inverters, Brushless DC motors, Stepper motors, and Brushed DC motors.

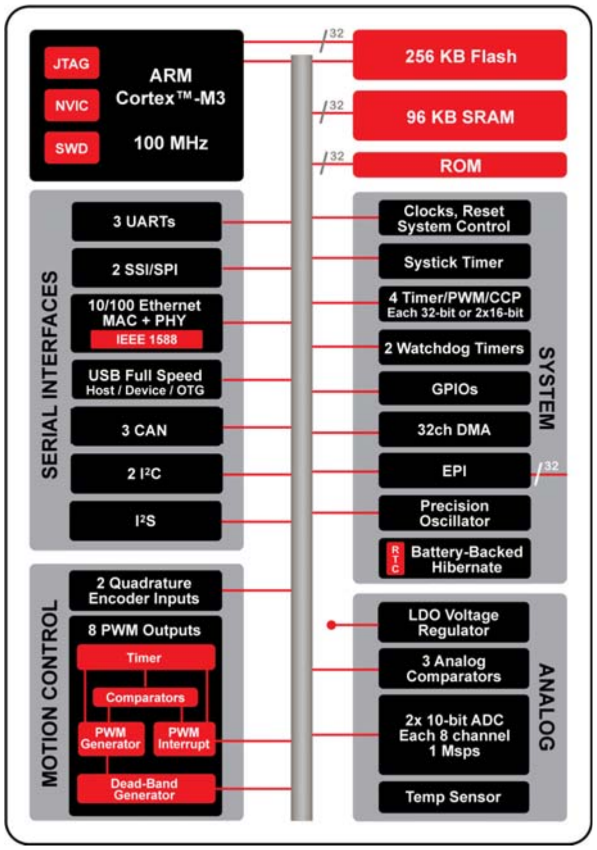


Figure 4. *Stellaris Family Block Diagram*

Other microcontrollers can struggle with an entire networked embedded control system. 8-bit and even 16-bit MCUs become all-engrossing when servicing requests from the network. Inadequate on-chip Flash can mean any significant Web page must be pulled from off-chip memory, often through serial ports. Small on-chip SRAM can thrash about keeping up with network communications demands. Even static Web pages have to be saved in external memory and be drawn in through serial ports because inadequate Flash is on-chip. In the meantime the main application might choke.

One of the easiest ways of embarking on a new project is by following an example. Whether it is determining the best motor control algorithm for the primary application, managing a CAN network, or putting together an embedded Web server so the new system can be checked and adjusted from across town, the evaluation kits for Stellaris MCUs are a valuable starting point. They come with fully-operational boards with example motors and sample programs using real control algorithms. A complete set of peripheral drivers are loaded on the boards. Complete user guides details application program interface (API) functions and programming examples giving easy access to the on-chip peripherals without having extensive deciphering of the data sheet. Software examples for TCP/IP stacks are given and the Web server is demonstrated. Essentially all software provided is available for study or modification.

An array of hardware and software development tools are available, including third-party compilers, debuggers, RTOSs, and design support from the vast ARM community.

Beyond the formal resources, a large user community is eager to share their knowledge.

Summary

An embedded web server is a great way to provide status, operation, production, and maintenance information about an embedded control system to an operator that is standing at the equipment or a thousand miles away. The system can be adjusted, reconfigured, and controlled using the same, familiar, Web-based interface. The detail and display of the information provided can be as rich as desired, yet the interface is largely the same and adds almost nothing to the cost of the equipment. Web servers are a bonus to applications that are already taking advantage of being networked, perhaps for coordination between machines, factories, or the front office. And access to the Internet can expand options that would be unfathomable in other scenarios.

Network access can be added to any embedded application without requiring additional circuitry, programming, memory, or performance. Proper selection of the microcontroller will make sure the main application isn't impacted by this system administration.

High performance microcontrollers can perform very sophisticated operations. Powerful, peripherals dedicated to motion control expand the control capabilities of the processor even more. A microcontroller family like Stellaris that is designed to also serve the networking needs of an embedded application can provide a balanced set of resources to make networking and the Web server on top of it run as smooth as the robot arms of the primary application. And everything integrated nicely together keeps costs to a minimum.

The system designer doesn't want to be burdened with learning a new processor instruction set, implementation of complex algorithms, and the ins and outs of networking protocol. Evaluation kits that demonstrate a complete working system similar to the end application, provided in a way that it is easily understood and modified, along with a good set of development tools and reliable support, enable engineers to put together advanced applications even with limited in-depth expertise in areas like motor control, networking, and Web page design.

Texas Instruments • 108 Wild Basin, Suite 350 • Austin, TX 78746
http://www.ti.com/stellaris

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps