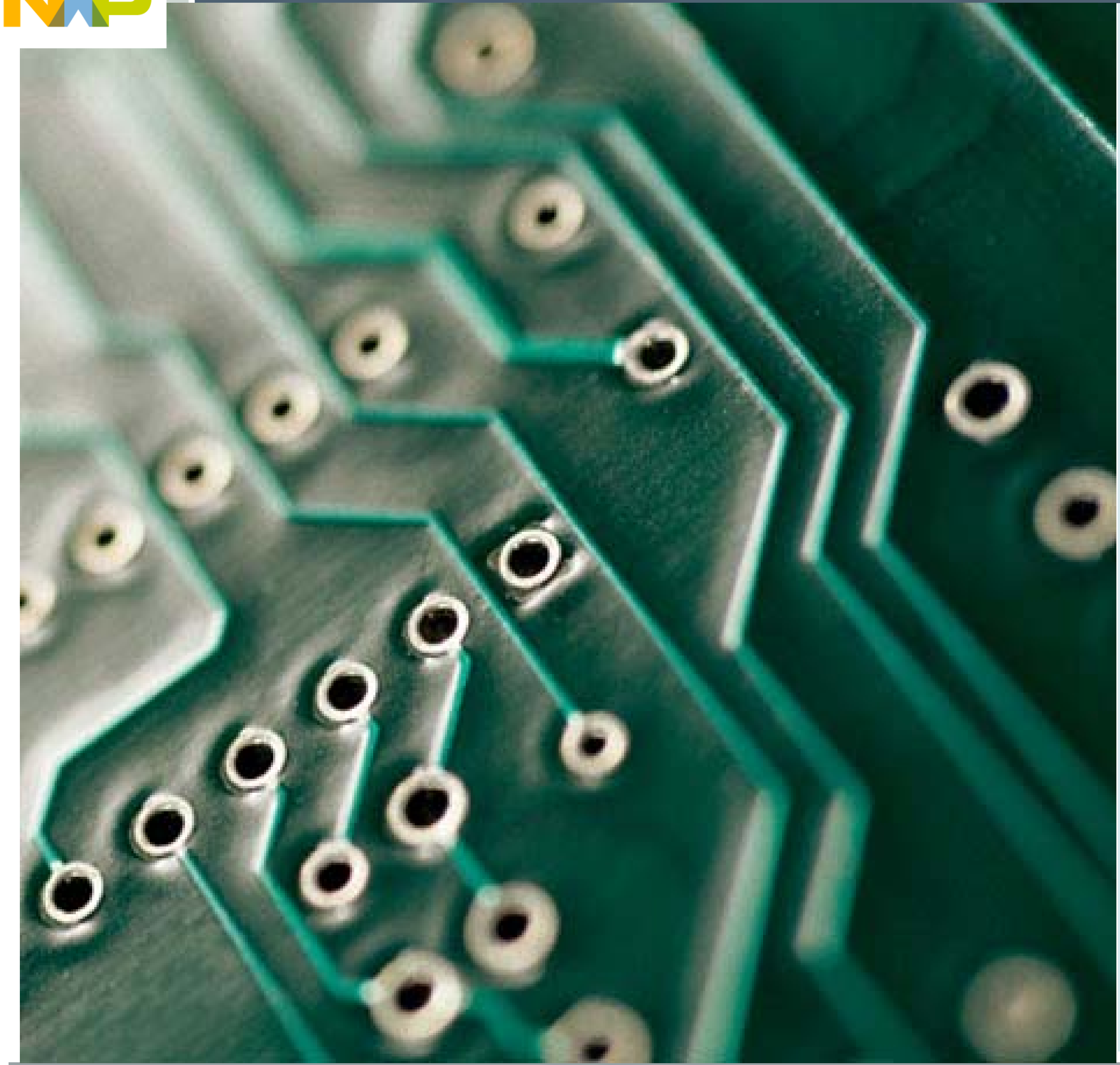
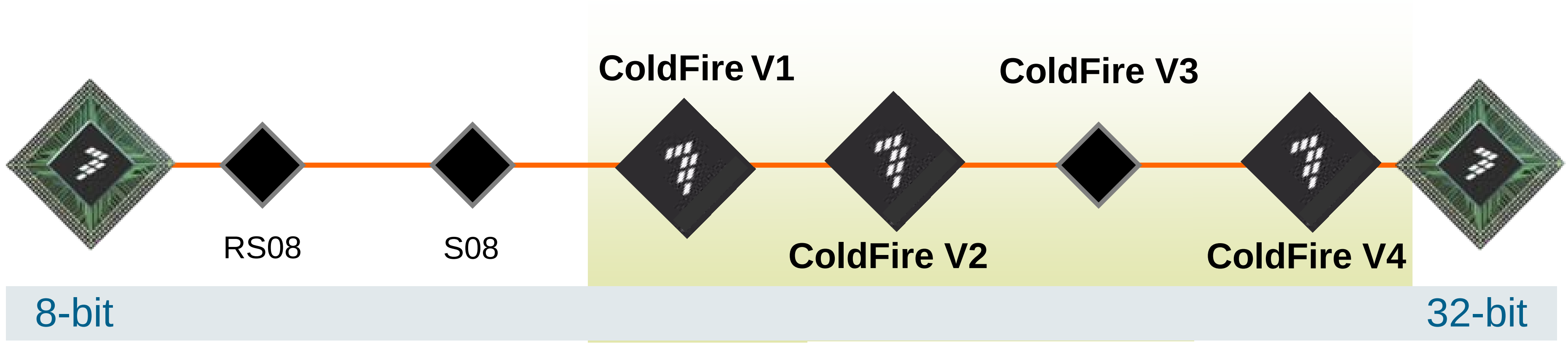
ColdFire® TCP/IP Stack Deep Dive



MCF5223x ColdFire® Ethernet MCU Family Overview

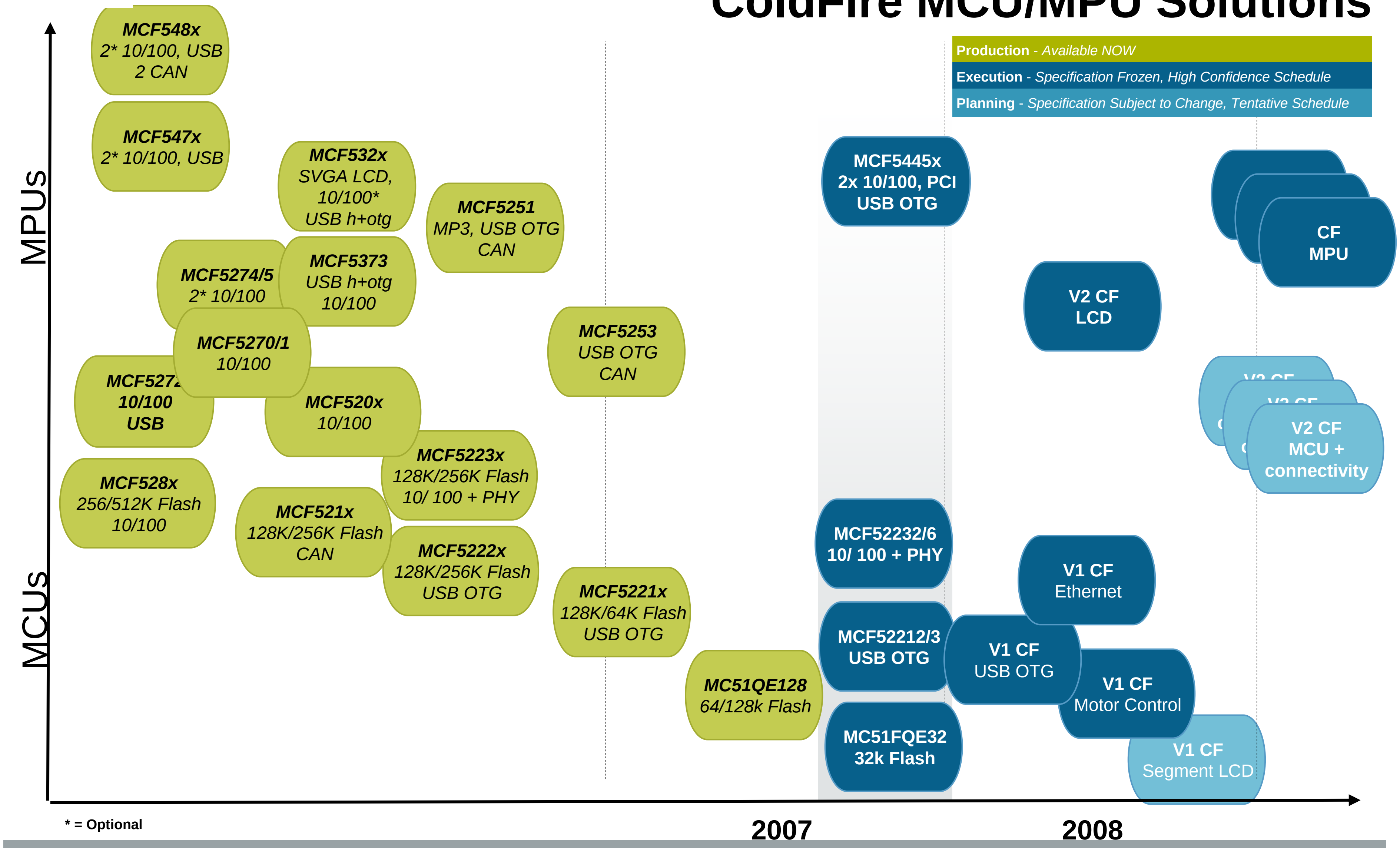
Enabling Ethernet Connectivity

The Controller Continuum. Only from Freescale.



- The Freescale Controller Continuum is the industry's first and only roadmap for compatible 8-bit and 32-bit architectures.
- From the ultra-low-end RS08 to the highest-performance ColdFire V4 devices, the Controller Continuum offers compatibility across our portfolio of consumer and industrial microcontrollers and microprocessors.
- Development tools such as CodeWarrior® for ColdFire and Processor Expert™ simplify migration and upward compatibility.

ColdFire MCU/MPU Solutions



* = Optional

5223x *ColdFire*® with Embedded Flash & Ethernet MAC&PHY

MCF5223x *ColdFire* Family

Targeted at Industrial Control Applications

- Environmental Monitoring
- Remote Data Collection
- Medical Pumps and Monitors
- Power-over-Ethernet
- Security/Access Panels
- Lighting Control Nodes
- Vending Machines

Key Features

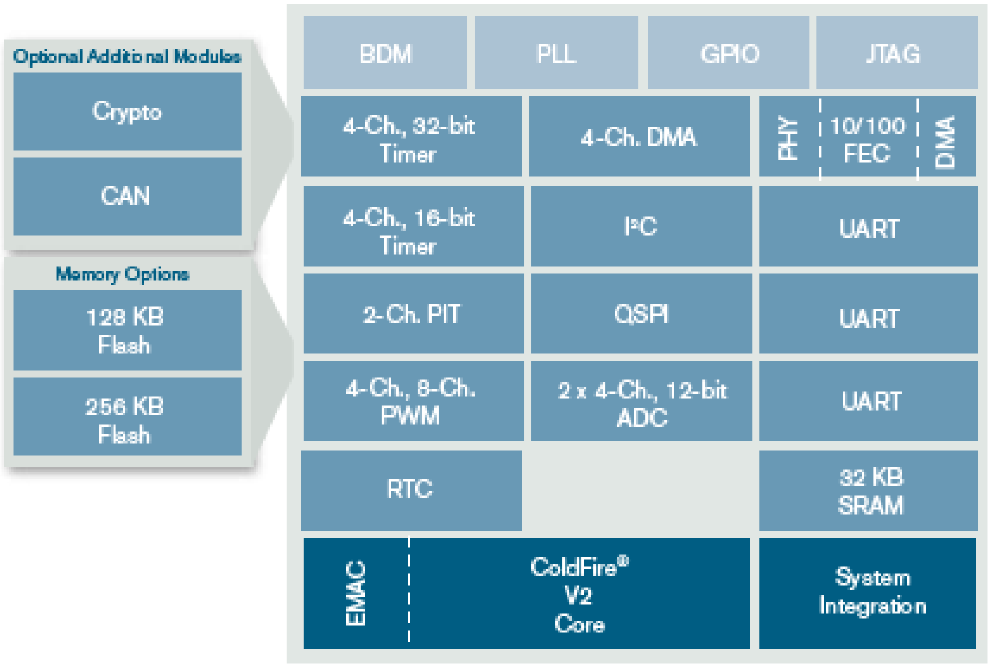
- 57 MIPS V2 Core with Enhanced Multiply and Accumulate for DSP-like functionality!
- Integrated Connectivity including:
  - **10/100 Ethernet Controller**
  - **10/100 Ethernet Physical Layer**
  - **CAN 2.0B Controller**
  - **Cryptographic Acceleration Unit**
- Additional control features include:
  - Up to 73 General Purpose I/O
  - 4ch. 32-bit timers with DMA support
- Starting from $7.99 suggested resale price

ColdFire® MCF52232, MCF52236
New Ethernet Devices

V2 ColdFire Core
- **Up to 46 Dhrystone 2.1 MIPS @ 50MHz**
- EMAC Module and HW Divide
    - *No external bus*

Memory
- 32KBytes SRAM
- Up to 256K bytes flash
    - 100K W/E cycles, 10 years data retention

Key Features
- **10/100 Ethernet MAC with PHY**
- Three (3) UARTs
- Queued Serial Peripheral Interface (QSPI)
- **I²C bus interface modules**
- 4 ch. 32-bit timers with DMA support
- 4ch. 16-bit Capture/Compare/PWM timers
- 2 ch. Periodic Interrupt Timer
- **8 ch. PWM timer with enhanced DAC capabilities**
- **4 ch. DMA controller**
- **Watchdog timer**
- **Real Time Clock (RTC) with 32kHz Oscillator**
- 8 ch. 12-bit A-to-D converter with Simultaneous Sampling
- **Up to 56 General-Purpose I/O**
- MCF52236 Starting from $5.39 at 10k resale with 256K Flash
- MCF52232 Starting from $4.94 at 10k resale with 128K Flash

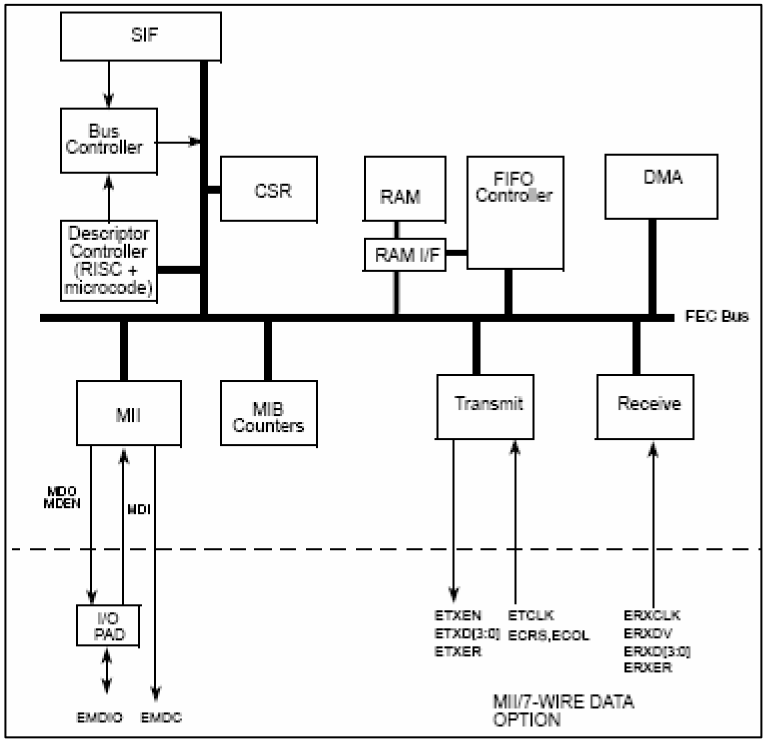


Single 3.3V Power Supply
Temp Range: -40°C to +85°C; 0°C to +70°C
80 LQFP

Freescale Confidential Proprietary

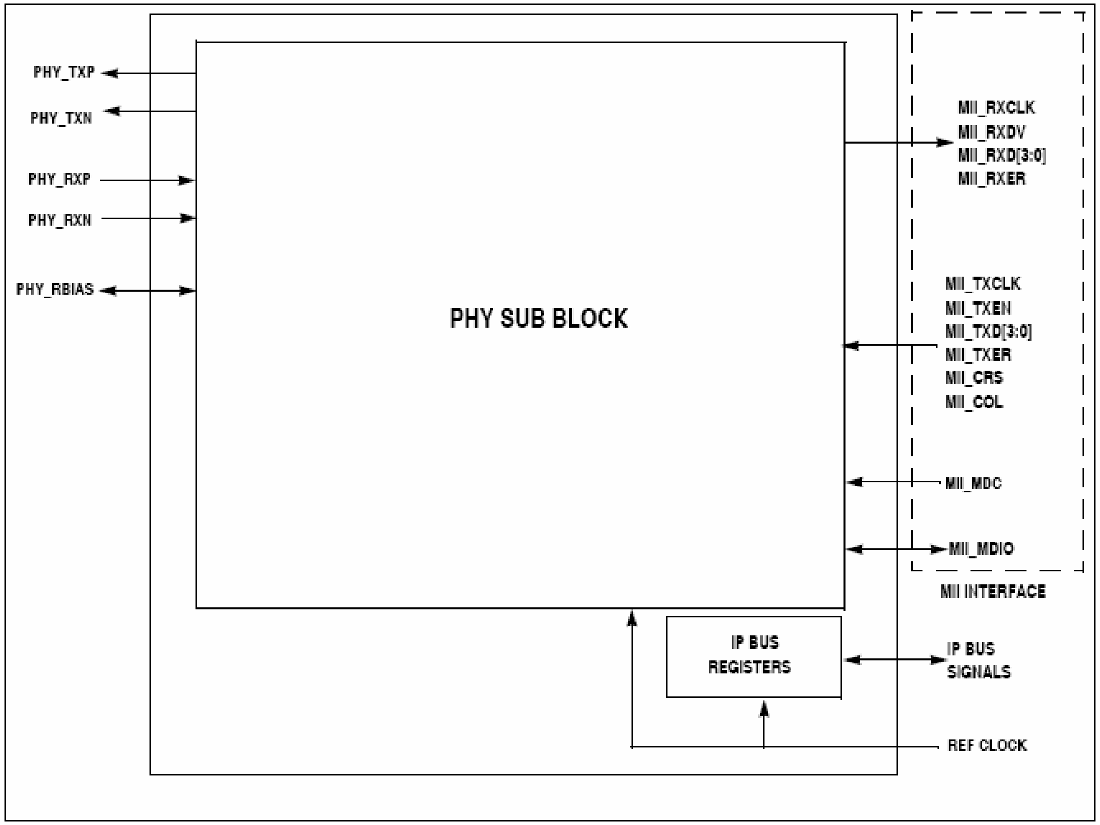
MCF5223x – Ethernet Media Access Controller (MAC)

- The Ethernet MAC supports 10/100 Mbps Ethernet/IEEE 802.3 networks
- IEEE 802.3 full duplex flow control
- Support for full-duplex operation (40Mbps throughput) with a minimum system clock rate of 50MHz
- Support for half-duplex operation (20Mbps throughput) with a minimum system clock rate of 25MHz



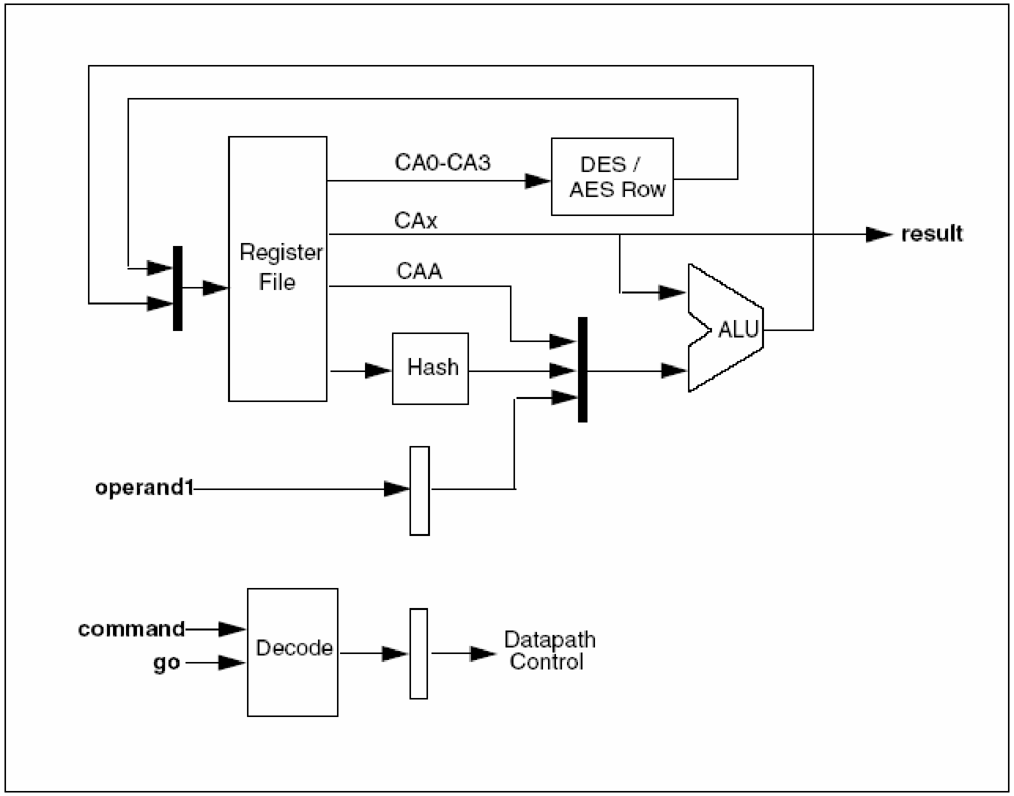
MCF5223x - ePHY

- The ePHY (embedded PHYsical layer interface) is IEEE 802.3 compliant
- Supports both the media-independent interface (MII) and the MII management interface
- Full-/half-duplex support in all modes
- Requires a 25-MHz crystal for its basic operation
- Supports Loopback modes



MCF5223x - Cryptographic Acceleration Unit (CAU)

- Uses standard ***ColdFire®*** coprocessor interface and instructions
- Simple, flexible programming model
- Supports DES, 3DES, AES, MD5 and SHA-1.
- Architecture allows for future enhancements
- Supports all ***ColdFire®*** cores



FlexCAN – Controller Area Network

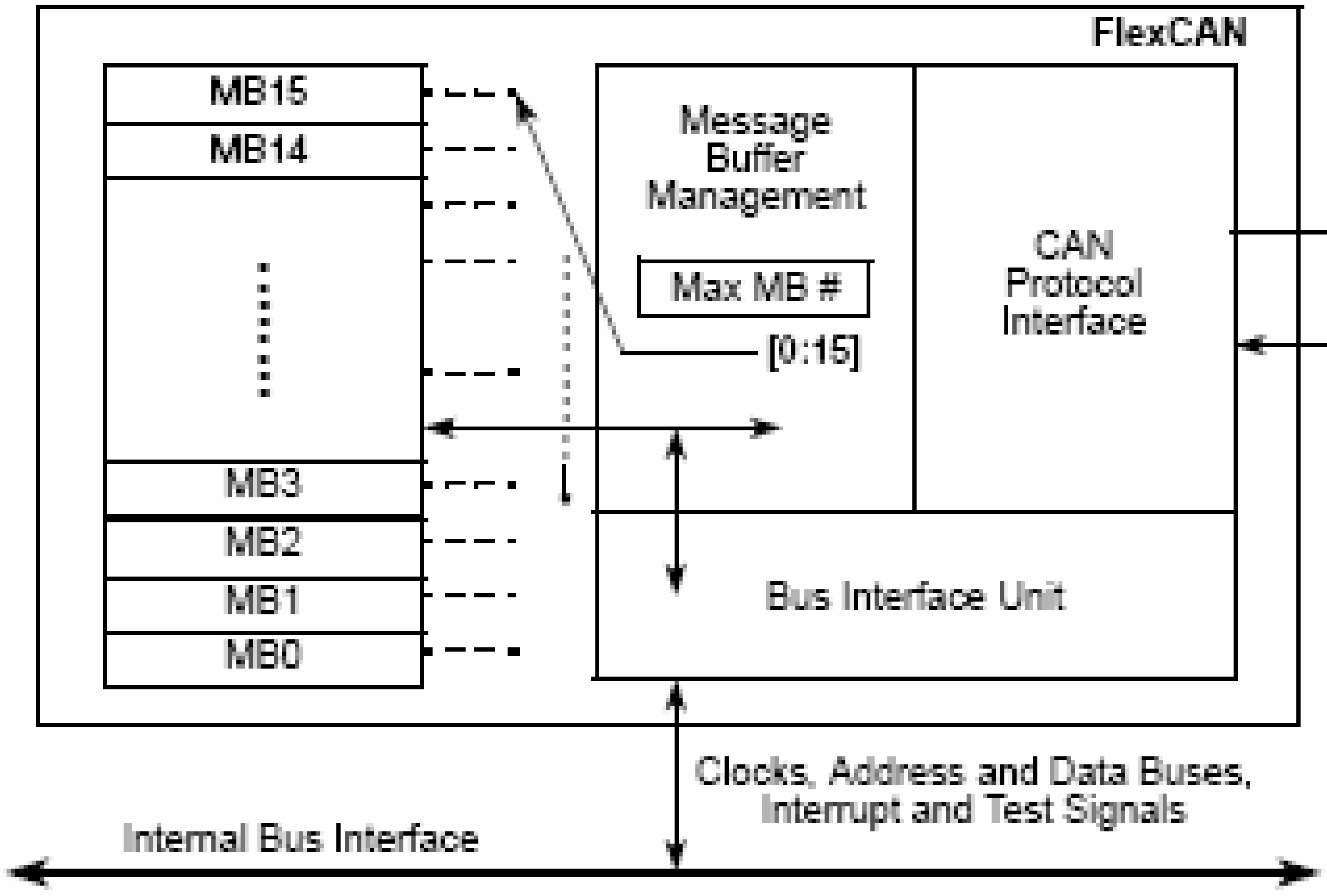


Figure 30-1. FlexCAN Block Diagram and Pinout

M52235EVB Evaluation Board

M52235EVB Evaluation Board and Development System

- Evaluation board with fully functional Power over Ethernet circuitry. Supports plug-in Zigbee daughter card
- Kit to include CD ROM, Power Supply, P&E BDM Cable, and Ethernet Crossover Cable
- Target Suggested Resale Price: $299

M52235EVB Software Support

- Free ColdFire_TCP/IP_Lite stack
- Free CodeWarrior® SPECIAL EDITION Included in Each Development Kit
- ColdFire Init – Graphical Initialization Tool

M52233DEMO Low cost demo board

M52233DEMO Low Cost Board

- Evaluation board with Plug-in Zigbee daughter card
- Kit to include CD ROM, Power Supply, and Ethernet Crossover Cable
- Target Suggested Resale Price: $99
- Available: May 2006

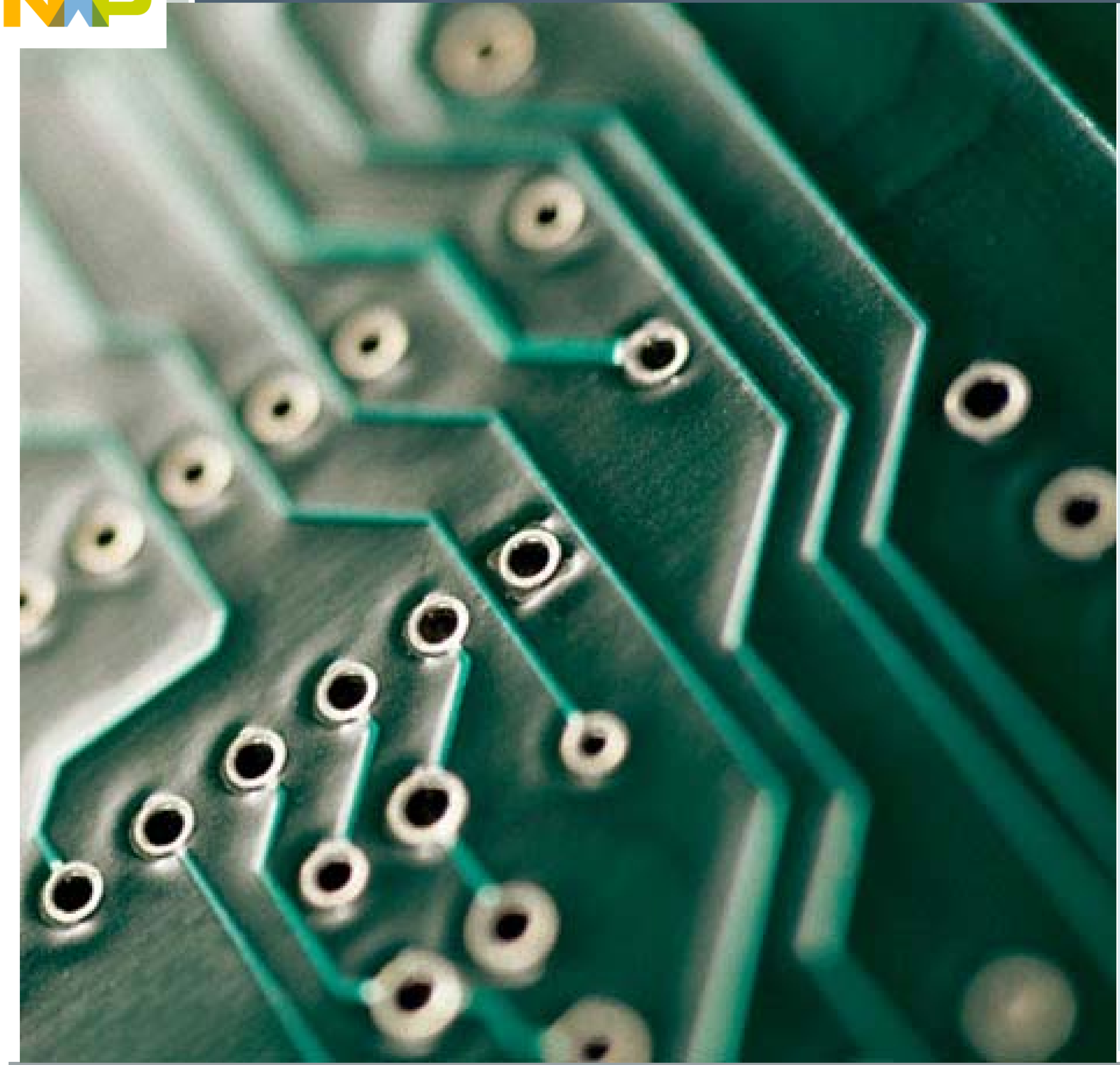
M52233DEMO Software Support

- Free ColdFire_TCP/IP_Lite stack
- Free CodeWarrior® SPECIAL EDITION Included in Each Development Kit
- ColdFire Init – Graphical Initialization Tool
-

EVB and DEMO Comparison

- MCF52235
  - 32K RAM 256K Flash, Ethernet with PHY, CAN, Crypto
  - 112 LQFP pin
- Light Sensor
- PoE capabilities
- 3 UARTs
- Supports plug-in Zigbee daughter card

- MCF52233
  - 32K RAM 256K Flash, Ethernet with PHY
  - 80 LQFP pin
- Accelerometer (3 axis g sensor)
- 1 UART
- Supports plug-in Zigbee daughter card



ColdFire TCP/IP LITE Stack

ColdFire_TCP/IP_Lite Stack

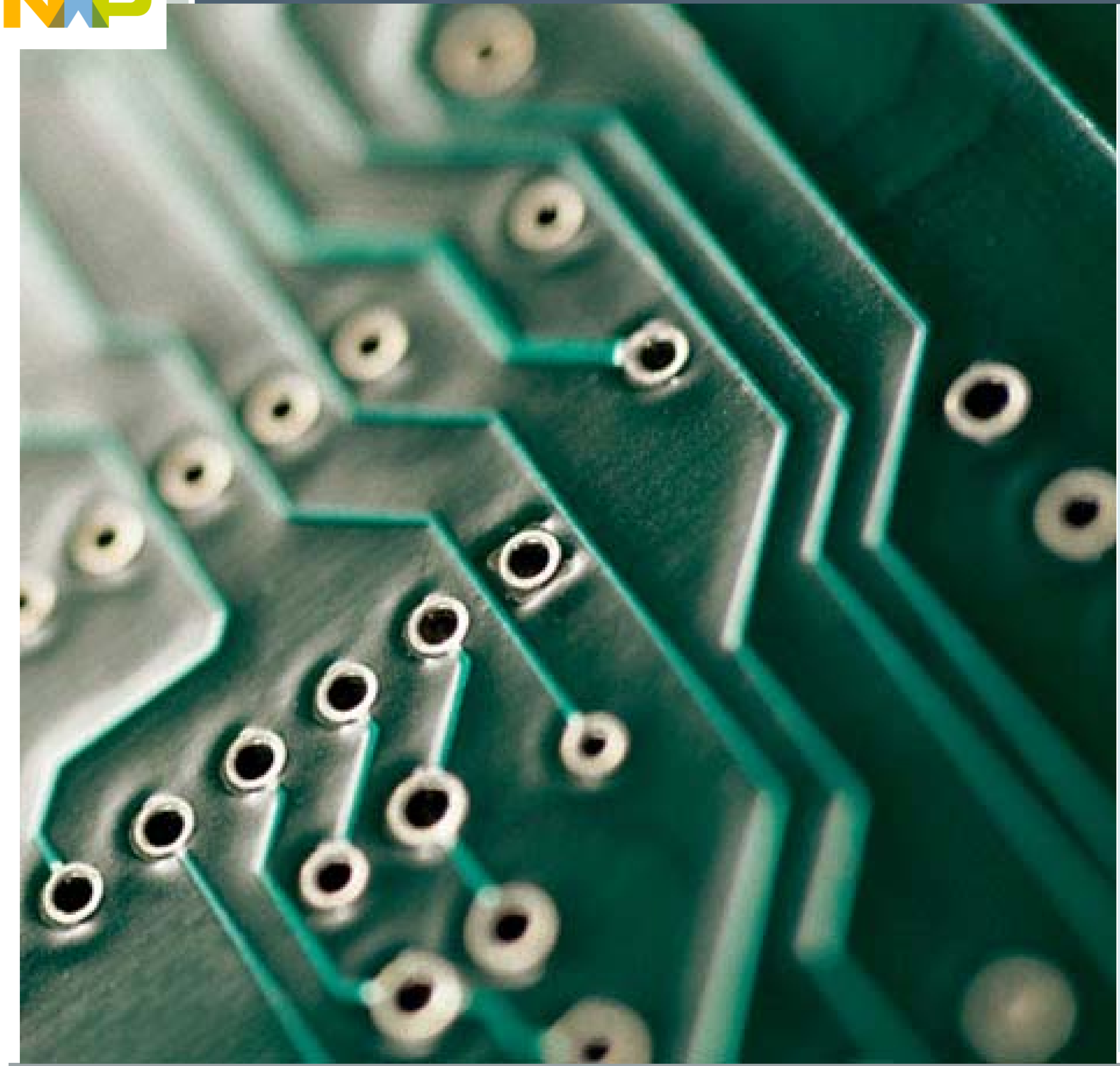
Available from Freescale:
InterNiche Technologies and Freescale have collaborated to provide an OEM version of InterNiche's NicheLite™, ColdFire_TCP/IP_Lite

Features

- Address Resolution Protocol (ARP)
- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)
- Dynamic Host Configuration Protocol
- (DHCP) Client
- Bootstrap Protocol (BOOTP)
- Trivial File Transfer Protocol (TFTP)

Freescale Provided additional free software:

- Web Server with Flash File System
- Serial to Ethernet
- Sample TCP, UDP, clients and servers.



Additional Features from InterNiche

NicheStack Products

These features can be purchased from Interniche as add-ons

SLIP
- Serial Line IP – Used to communicate IP over a modem.

PPP
- Point to Point Protocol – Used to establish a connection with a ISP.

SMTP client
- Simple Mail Transport Protocol – Used to send a email message.

SNMP
- Simple Network Management Protocol – Used to exchange Node information.

DHCP server
- Dynamic Host Configuration Protocol – The server is used to assign IP addresses in a network.

Telnet Server
- Provides a method of “logging” onto the node from a remote location.

DNS server
- Dynamic Name Server – Used to translate URL names to a IP address.

FTP client
- File Transfer Protocol client – Used to transfer files to and from the device.

FTP server
- File Transfer Protocol server – Used to allow the transfer of files to and from the device.

SSL
- Secure Socket Layer – Provides a encrypted connection between to devices.

Web Server
- Or HTTP server – Used to send web pages to a web browser client.

IPv6	IPv4	v4/v6	Lite	
	✓	✓		Auto-IP
✓	✓	✓	✓	DDNS
	✓	✓	✓	DHCP Server
✓	✓	✓	✓	DNS Client
	✓	✓		DNS Server
✓	✓	✓	✓	Email Alerter
✓	✓	✓	✓	FTP Server + Client
	✓	✓		IP Multicast
✓	✓	✓		IPSec/IKE
	✓	✓		NAT
✓	✓	✓		NAT-PT
✓	✓	✓	✓	POP3
✓	✓	✓	✓	PPP
✓	✓	✓	✓	▪ MS-CHAP
✓	✓	✓		▪ MultiLink
✓	✓	✓		▪ PPPoE
	✓	✓		RIP
	✓	✓		SSL/TLS
✓	✓	✓	✓	SNMPv1
✓	✓	✓	✓	SNMPv2(c)
✓	✓	✓	✓	SNMPv3
✓	✓	✓	✓	Telnet Server
✓	✓	✓	✓	TFTP Client+Server
✓	✓	✓	✓	Web Server
✓	✓	✓	✓	▪ HTML Compiler
	✓	✓		▪ Web Server-SSL
✓	✓	✓	✓	NicheTool

NicheStack PPP

Supports:

- LCP
- IPCP

Hayes dialing code

VJ Header Compression

PAP, CHAP and MS-CHAP security

Multiple Simultaneous Links

Supports DHCP

Compliant with RFC’s 1144, 1332, 1344, 1661, 1662 and 1994

PPPoE - PPP over Ethernet

Allows PPP connections through Ethernet adapters

- Used by broadband service providers to allow PPP authentication
- Maintains the familiar "dial-up experience" when connecting with a broadband modem
- Operates between PPP and Ethernet driver
- RFC 2516

MultiLink PPP

A method for splitting, recombining and sequencing datagrams across multiple logical data links.

Based on an LCP option negotiation.

Originally designed for multiple bearer channels in ISDN

RFC 1990

Secure Sockets Layer Library

Provides Secure Sockets Layer for embedded web services

Layer of security for HTTP web traffic between client and server

Based on public key asymmetric cryptography

Requires upgrade to NicheStack – won’t work with NicheLite/FreescaleLite

NicheStack SSL -- Features

Adds SSL to web services

- NicheStack
- WebPort - HTTP Server
- NicheView Browser

RSA key exchange method with 1024 bit key generation and Triple DES encryption

Blocking and non-blocking modes

Supports IETF

- SSL v2.0
- SSL v3.0
- TLS v1.0

Includes

- SSL Library
- API Library for server-side SSL
- Static Library

Based on Open SSL

NicheStack SSL -- Benefits

Authenticated, encrypted communication

NicheStack SSL protects the integrity of the embedded device and its configuration

SNMP

Agent

Uses UDP/IP

MIB-database

Variables

- Statistic values about communication
- Private extensions
- Access rights configurable

MIB-compiler.

Advantages of InterNiche SNMP

Portable

MIB Compiler

- An automated tool to help support new MIBs

Sample Code

- implementation of MIB2, SNMPv3 RFCs

Complies with RFC standards

SMTP Email Alerter

Low cost mechanism for adding EMAIL reporting capability to embedded application.

- Supports Simple Mail Transfer protocol (SMTP)
- Sends predetermined messages from an embedded system to a local or remote email address
- Sockets interface makes porting quick and easy
- Supports multiple target email addresses
- Supports many individual messages and formats
- Compliant with RFC’s 821, 869 & 870

Telnet

- Provides network accessibility for remote configuration and monitoring
- Compatible with commercial TELNET Clients (Windows, NT UNIX, etc.)
- Supports multiple TELNET sessions
- Highly portable
- Small memory requirement
- Compliant with RFC 854

FTP Client/Server

Adds file server capabilities

Supports Passive mode

Multi user and multi session

Two Way Tasking - no special multitasking features are required

Run by polling from a central loop or take advantage of an RTOS suspend/resume feature

Can open sessions as a Server or Client

Compliant with RFC 959

NicheTool (included with NicheLite product)

Advanced Debugging and Tuning Suite

Included with NicheStack and NicheLite

Allows developers to rapidly view, debug and tune their target system

Reduces need for in-depth TCP/IP expertise

NicheTool Features

An expandable menu system with a command line interface (CLI)

Direct visibility into key networking structures

Access RFC 1213 MIB II statistics and approximately 100 extensions

Trace packets as they travel through stack layers

View buffer utilization

NicheTool Benefits

It allows engineers to quickly:

Verify the stack build

Verify hardware / network connections

Find errors and trace network connections

Optimize throughput and memory utilization

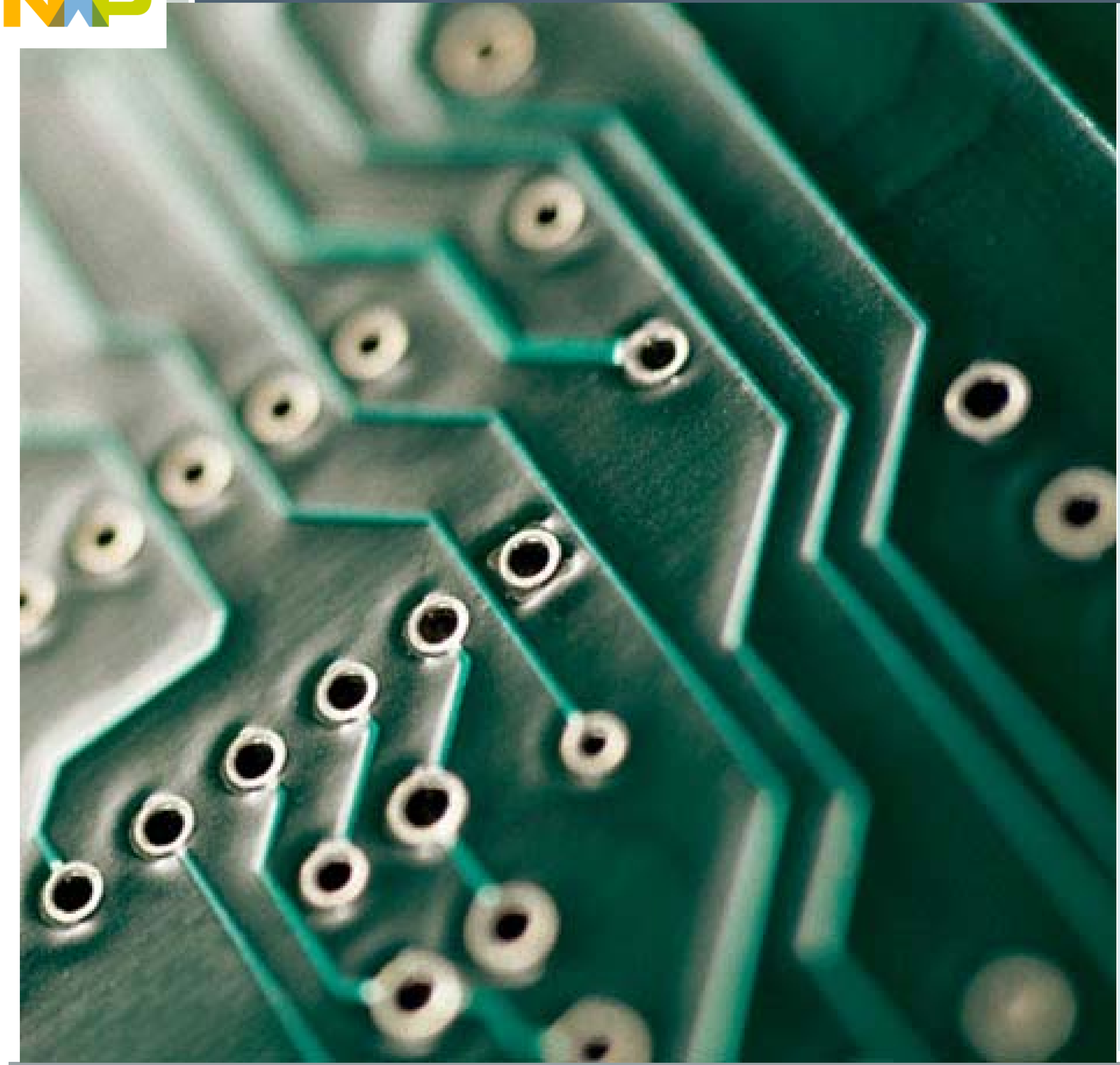
Create customized menus for the target system

InterNiche Prices

Have customers contact InterNiche for price quotes

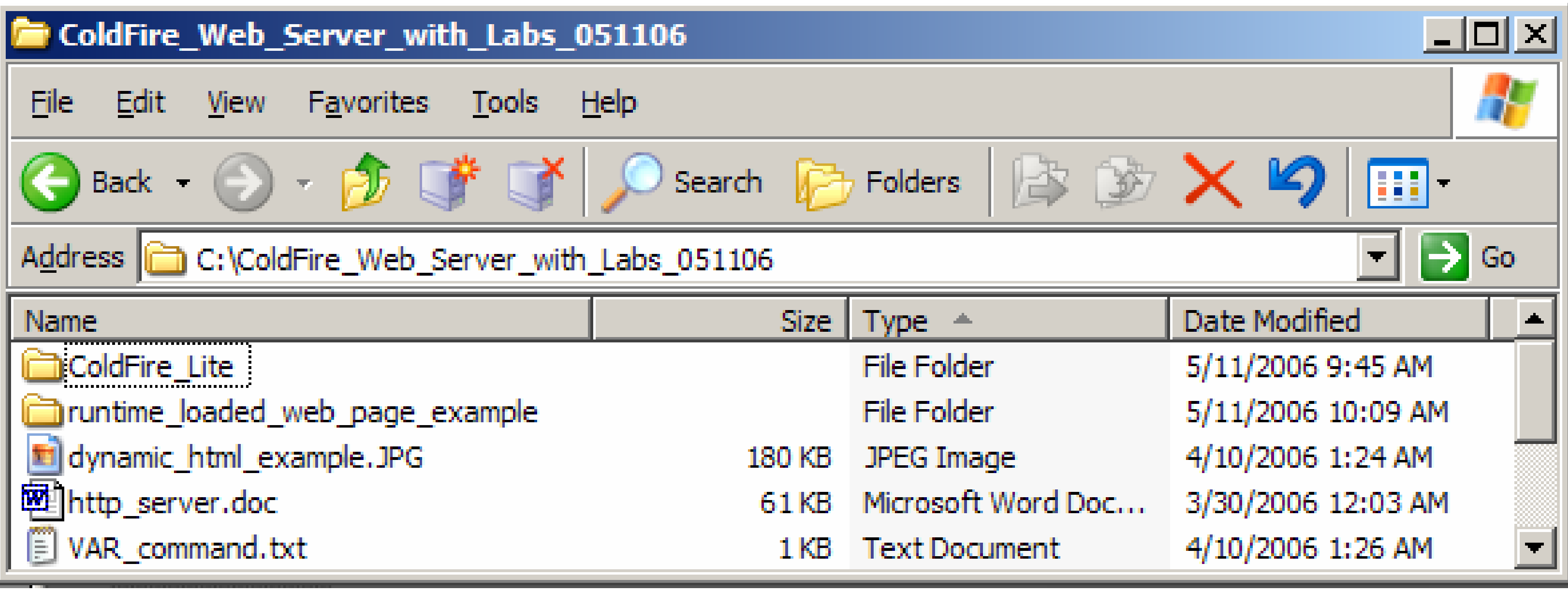
Single product license pricing

- SMTP $3000
- PPP $3000
- SNMP v1/v2c $5000
- FTP $2000
- SSL $10,000 (requires NicheStack which is $8000)



ColdFire TCP/IP Lite Project Overview

The Directory Structure


ColdFire_Web_Server_with_Labs_051106
File Edit View Favorites Tools Help
Back
Search
Folders
Address
C:\ColdFire_Web_Server_with_Labs_051106
Go

Name	Size	Type	Date Modified
ColdFire_Lite		File Folder	5/11/2006 9:45 AM
runtime_loaded_web_page_example		File Folder	5/11/2006 10:09 AM
dynamic_html_example.JPG	180 KB	JPEG Image	4/10/2006 1:24 AM
http_server.doc	61 KB	Microsoft Word Doc...	3/30/2006 12:03 AM
VAR_command.txt	1 KB	Text Document	4/10/2006 1:26 AM

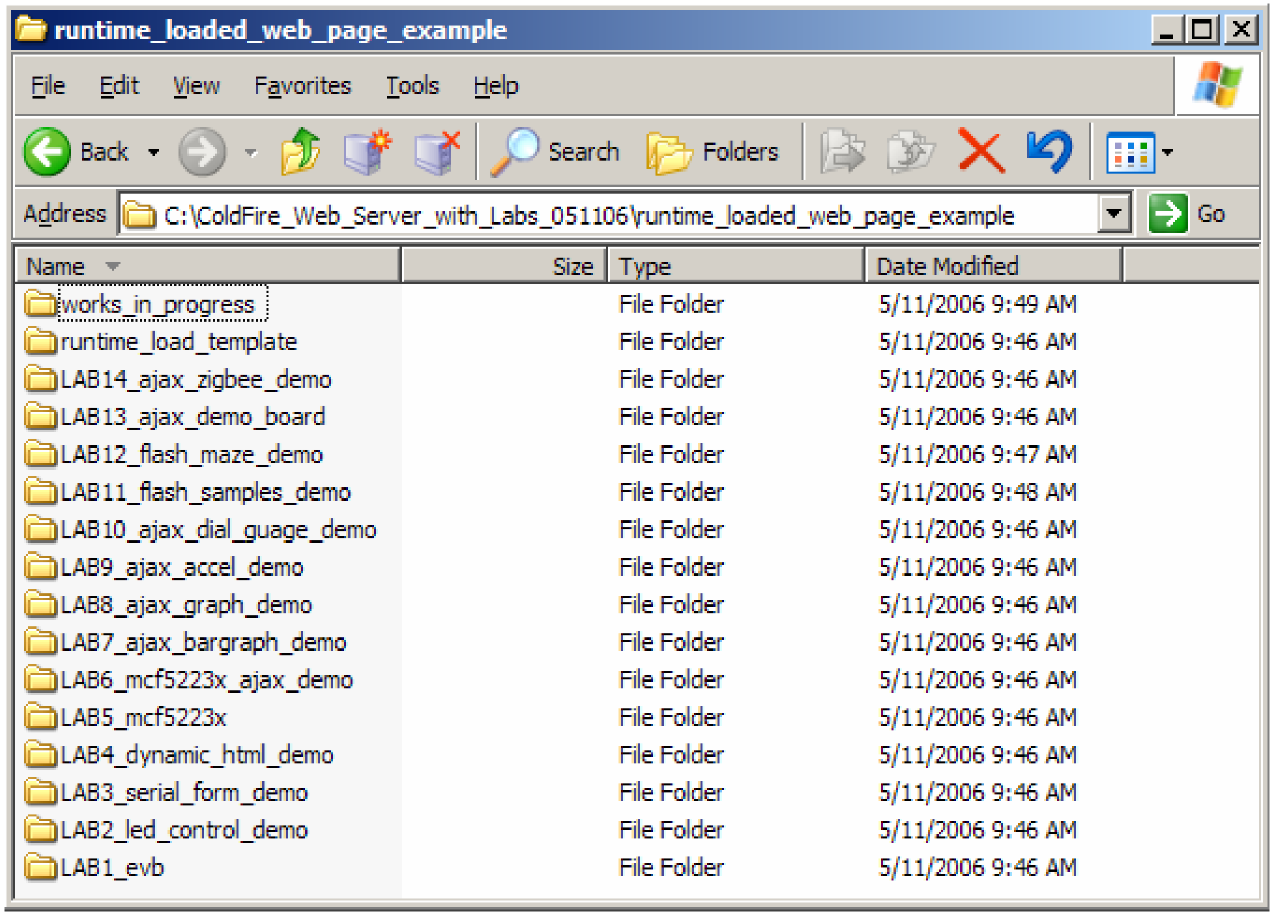
ColdFire_Lite = Interniche stack and projects

Runtime_loaded_web_page_example = Loadable labs

Directory Details – Runtime Loadable Demos/Labs

Runtime_loaded_web_page_example directory

This directory contains the runtime loadable demos/labs.



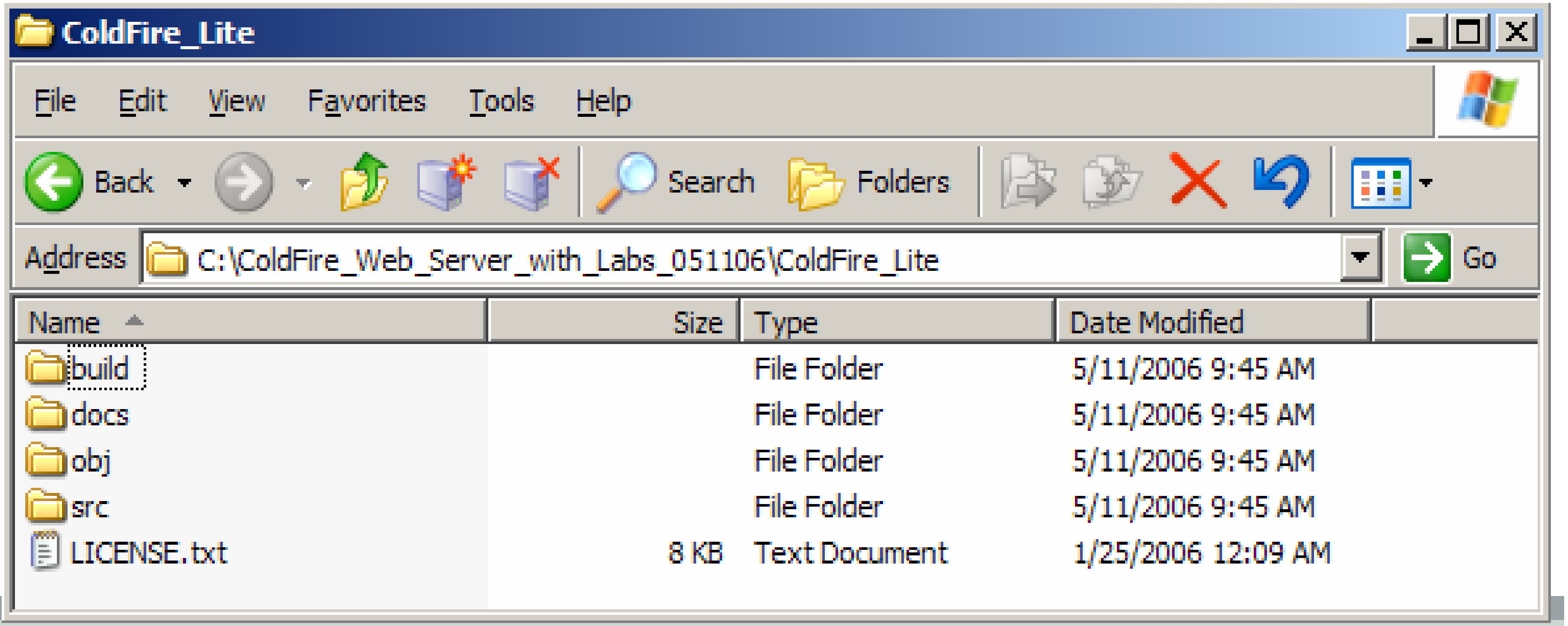
C:\ColdFire_Web_Server_with_Labs_051106\runtime_loaded_web_page_example

Name	Size	Type	Date Modified
works_in_progress		File Folder	5/11/2006 9:49 AM
runtime_load_template		File Folder	5/11/2006 9:46 AM
LAB14_ajax_zigbee_demo		File Folder	5/11/2006 9:46 AM
LAB13_ajax_demo_board		File Folder	5/11/2006 9:46 AM
LAB12_flash_maze_demo		File Folder	5/11/2006 9:47 AM
LAB11_flash_samples_demo		File Folder	5/11/2006 9:48 AM
LAB10_ajax_dial_guage_demo		File Folder	5/11/2006 9:46 AM
LAB9_ajax_accel_demo		File Folder	5/11/2006 9:46 AM
LAB8_ajax_graph_demo		File Folder	5/11/2006 9:46 AM
LAB7_ajax_bargraph_demo		File Folder	5/11/2006 9:46 AM
LAB6_mcf5223x_ajax_demo		File Folder	5/11/2006 9:46 AM
LAB5_mcf5223x		File Folder	5/11/2006 9:46 AM
LAB4_dynamic_html_demo		File Folder	5/11/2006 9:46 AM
LAB3_serial_form_demo		File Folder	5/11/2006 9:46 AM
LAB2_led_control_demo		File Folder	5/11/2006 9:46 AM
LAB1_evb		File Folder	5/11/2006 9:46 AM

Directory Details – ColdFire_Lite

ColdFire_Lite directory

The ColdFire_Lite directory contains the TCP/IP stack and Web Server Firmware.

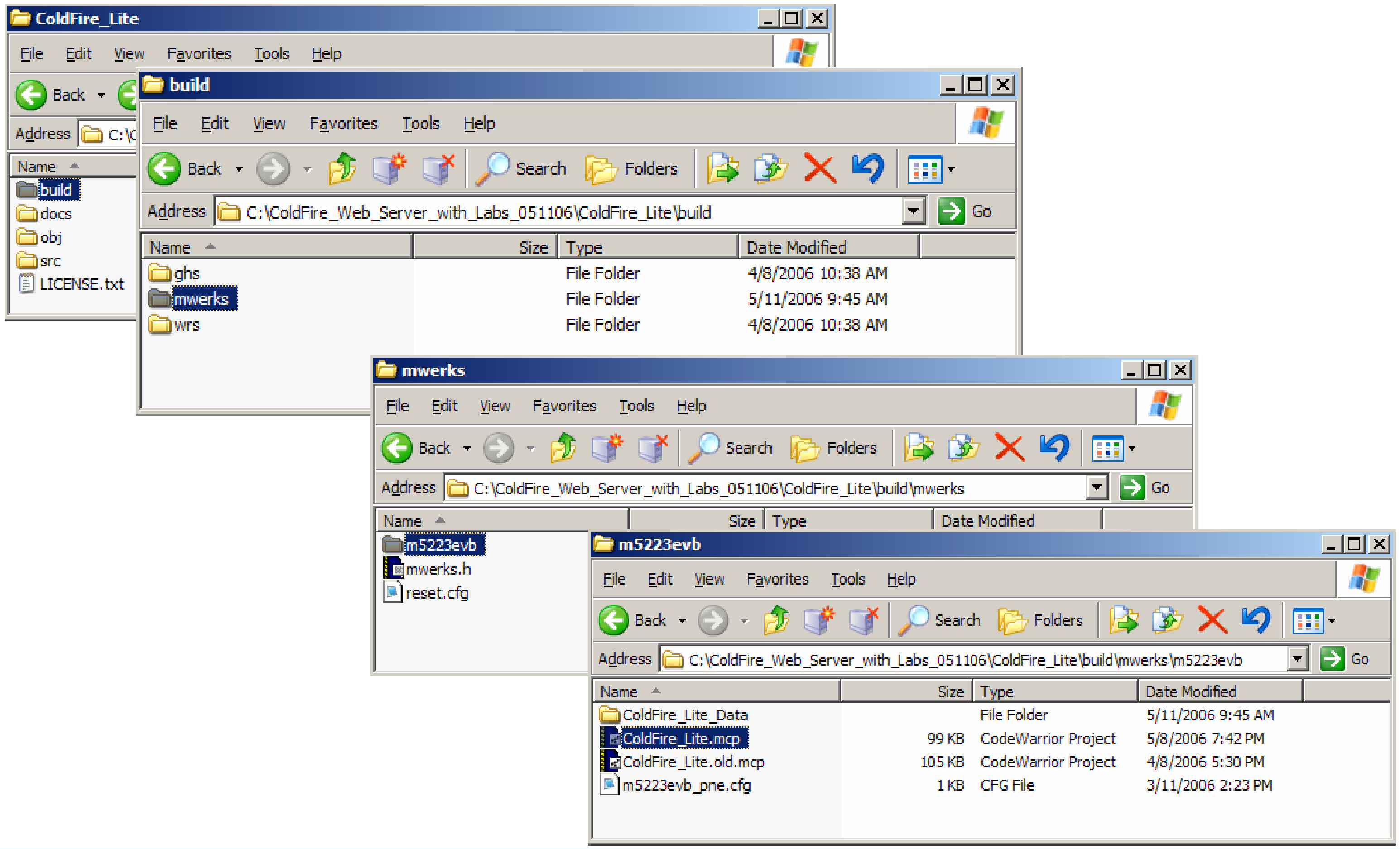


C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite

Name	Size	Type	Date Modified
build		File Folder	5/11/2006 9:45 AM
docs		File Folder	5/11/2006 9:45 AM
obj		File Folder	5/11/2006 9:45 AM
src		File Folder	5/11/2006 9:45 AM
LICENSE.txt	8 KB	Text Document	1/25/2006 12:09 AM

ColdFire_Lite Project File

The project File is used to open the project in CodeWarrior®.



ColdFire_Lite

Name
build
docs
obj
src
LICENSE.txt

build — Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build

Name	Size	Type	Date Modified
ghs		File Folder	4/8/2006 10:38 AM
mwerks		File Folder	5/11/2006 9:45 AM
wrs		File Folder	4/8/2006 10:38 AM

mwerks — Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build\mwerks

Name
m5223evb
mwerks.h
reset.cfg

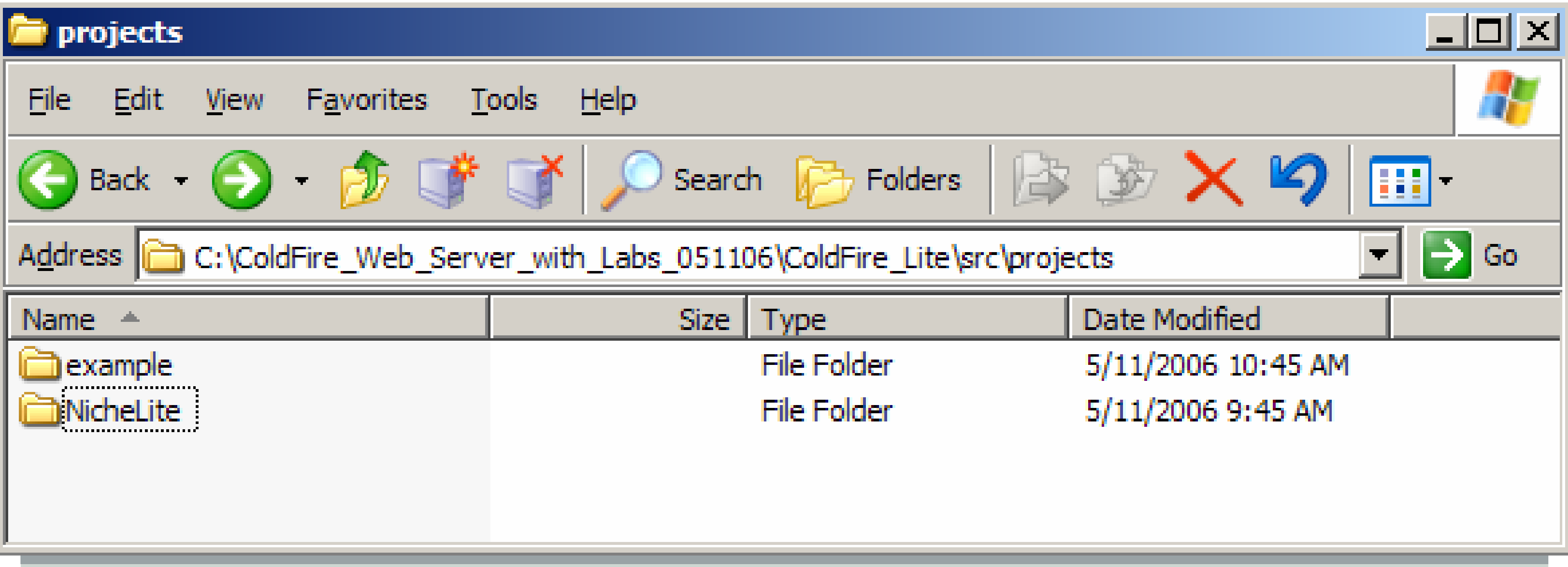
m5223evb — Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build\mwerks\m5223evb

Name	Size	Type	Date Modified
ColdFire_Lite_Data		File Folder	5/11/2006 9:45 AM
ColdFire_Lite.mcp	99 KB	CodeWarrior Project	5/8/2006 7:42 PM
ColdFire_Lite.old.mcp	105 KB	CodeWarrior Project	4/8/2006 5:30 PM
m5223evb_pne.cfg	1 KB	CFG File	3/11/2006 2:23 PM

The NichLite directory

The NicheLite directory contains the source to the TCP/IP stack.

ColdFire_Lite\src\projects



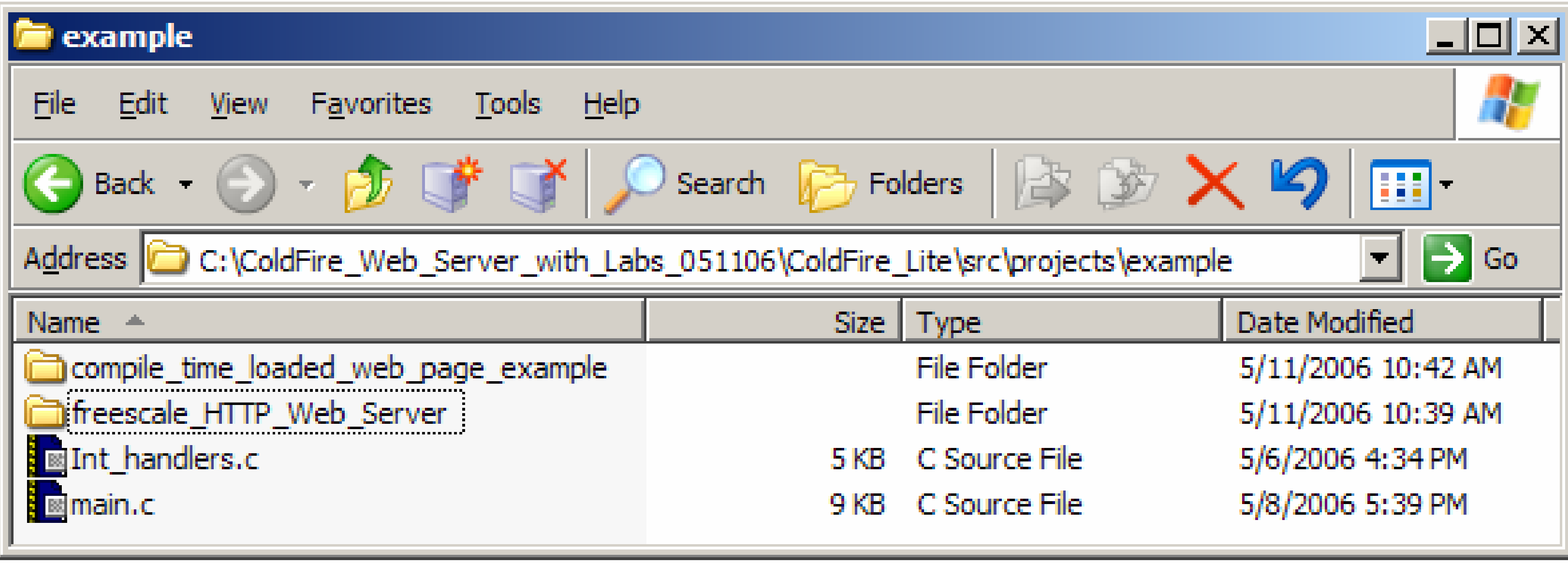
Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\src\projects

Name	Size	Type	Date Modified
example		File Folder	5/11/2006 10:45 AM
NicheLite		File Folder	5/11/2006 9:45 AM

Freescale_HTTP_Web_Server directory

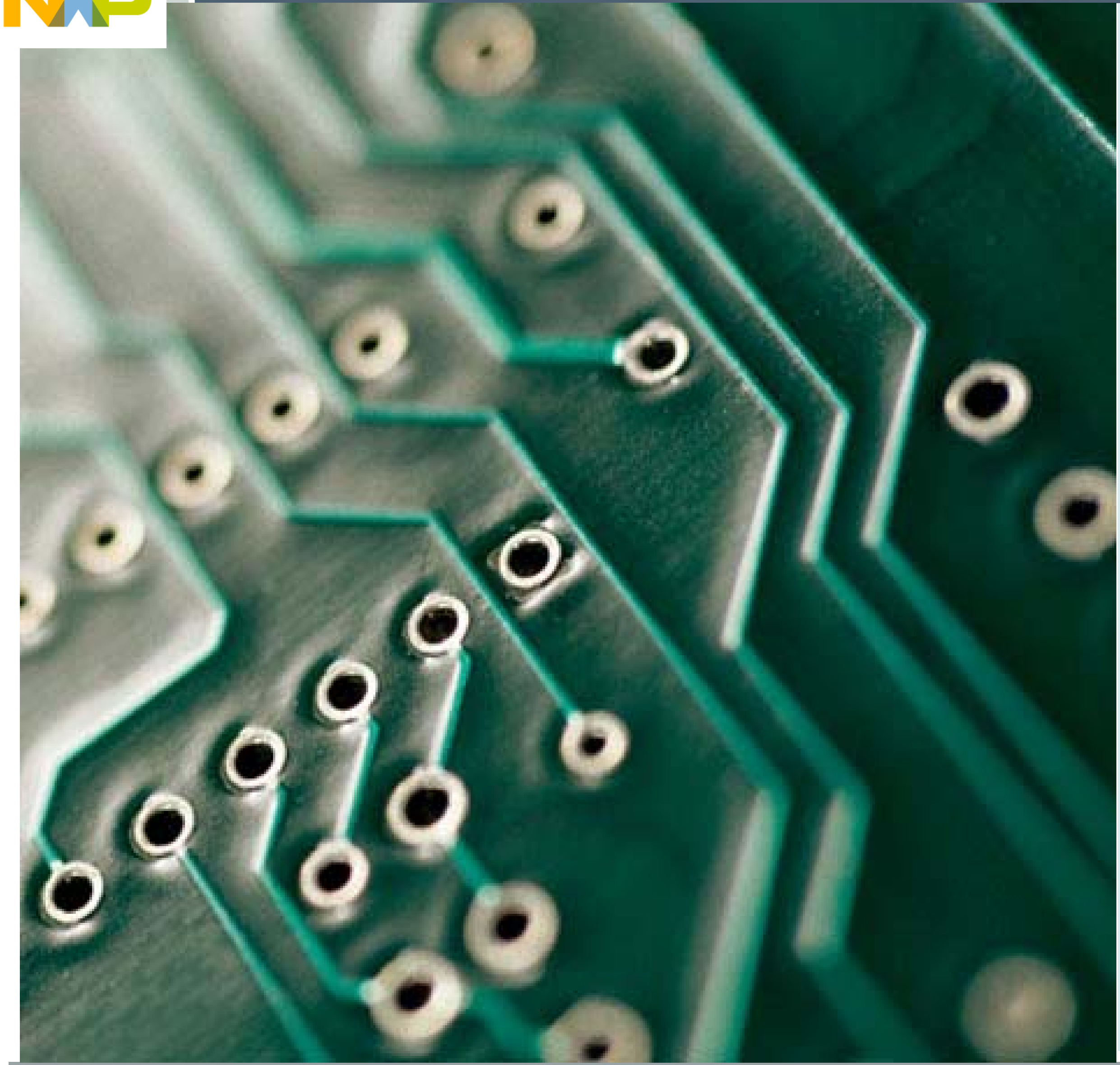
The Freescale_HTTP_Web_Server directory contains the source code for the Freescale Web Server.

ColdFire_Lite\src\projects\example



C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\src\projects\example

Name	Size	Type	Date Modified
compile_time_loaded_web_page_example		File Folder	5/11/2006 10:42 AM
freescale_HTTP_Web_Server		File Folder	5/11/2006 10:39 AM
Int_handlers.c	5 KB	C Source File	5/6/2006 4:34 PM
main.c	9 KB	C Source File	5/8/2006 5:39 PM



Opening ColdFire TCP/IP Lite

Locate and Open the TCP/IP/Web Server Project

Close all open CodeWarrior® Project Windows.

Choose File > Open

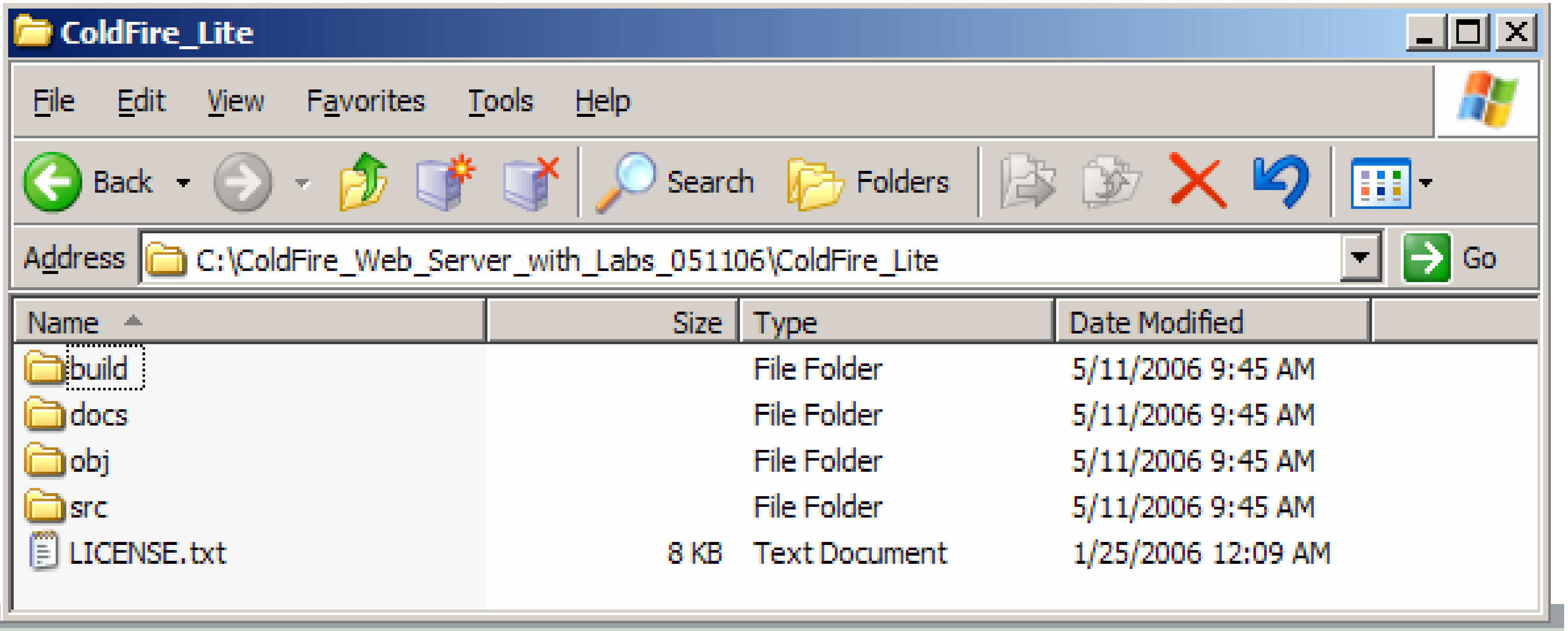
Browse to the ColdFire Lite Directory.

- This will be located where you unzipped the ColdFire Lite project, or if you are using a Freescale laptop

Directory Details – ColdFire_Lite

ColdFire_Lite directory

The ColdFire_Lite directory contains the TCP/IP stack and Web Server Firmware.

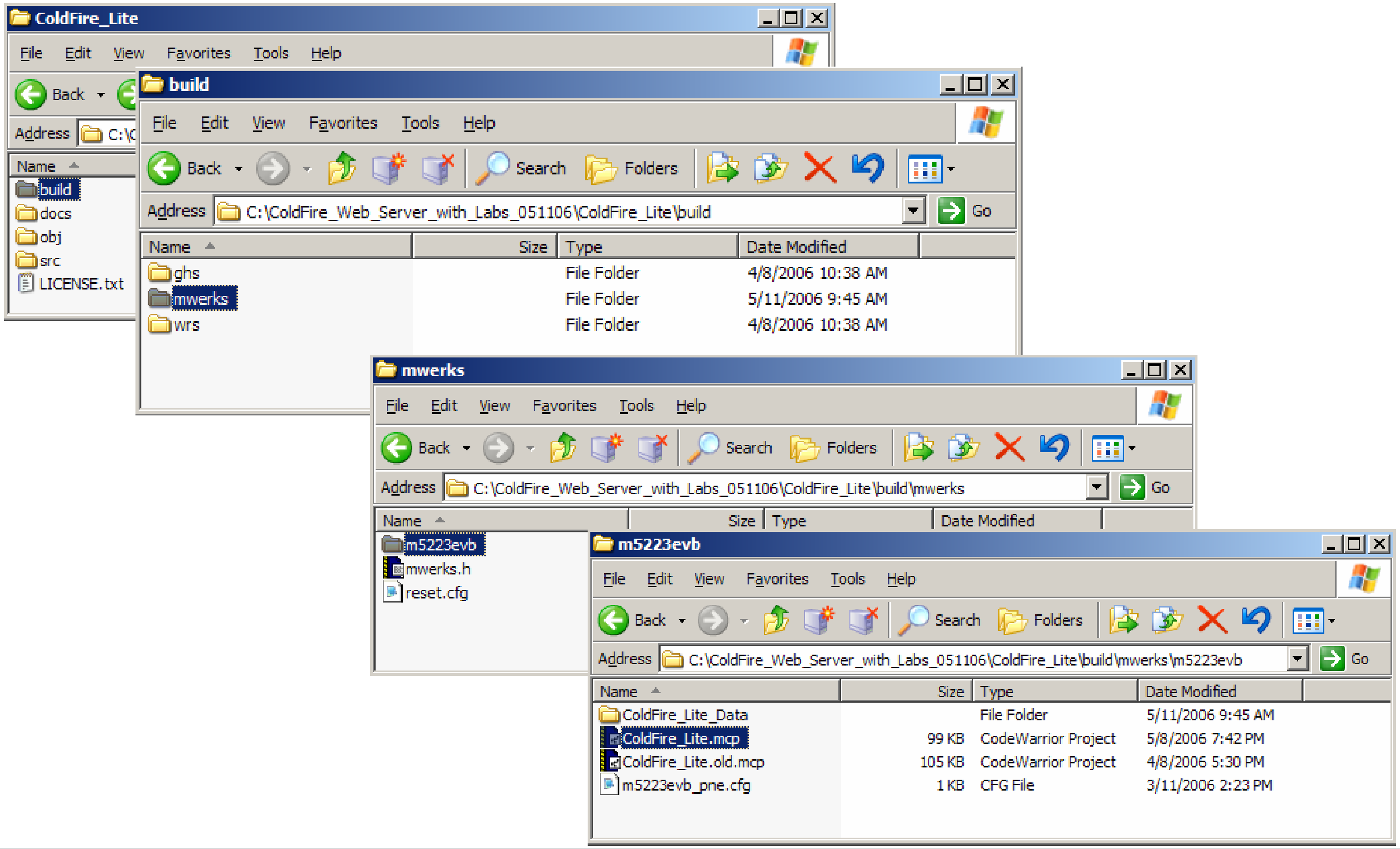


C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite

Name	Size	Type	Date Modified
build		File Folder	5/11/2006 9:45 AM
docs		File Folder	5/11/2006 9:45 AM
obj		File Folder	5/11/2006 9:45 AM
src		File Folder	5/11/2006 9:45 AM
LICENSE.txt	8 KB	Text Document	1/25/2006 12:09 AM

ColdFire_Lite Project File

The project File is used to open the project in CodeWarrior®.



Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build

Name	Size	Type	Date Modified
ghs		File Folder	4/8/2006 10:38 AM
mwerks		File Folder	5/11/2006 9:45 AM
wrs		File Folder	4/8/2006 10:38 AM

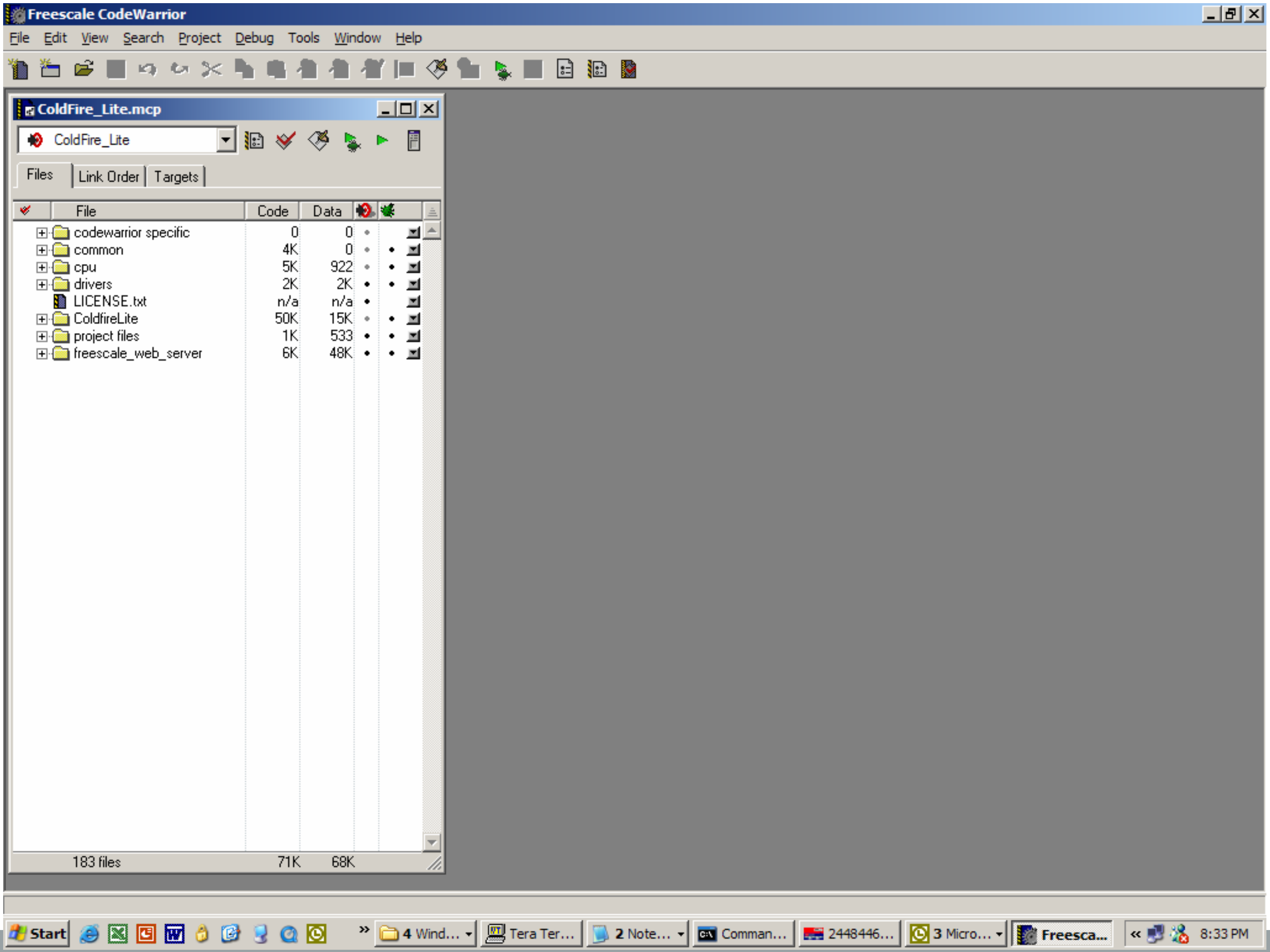
Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build\mwerks

Name	Size	Type	Date Modified
m5223evb			
mwerks.h			
reset.cfg			

Address: C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\build\mwerks\m5223evb

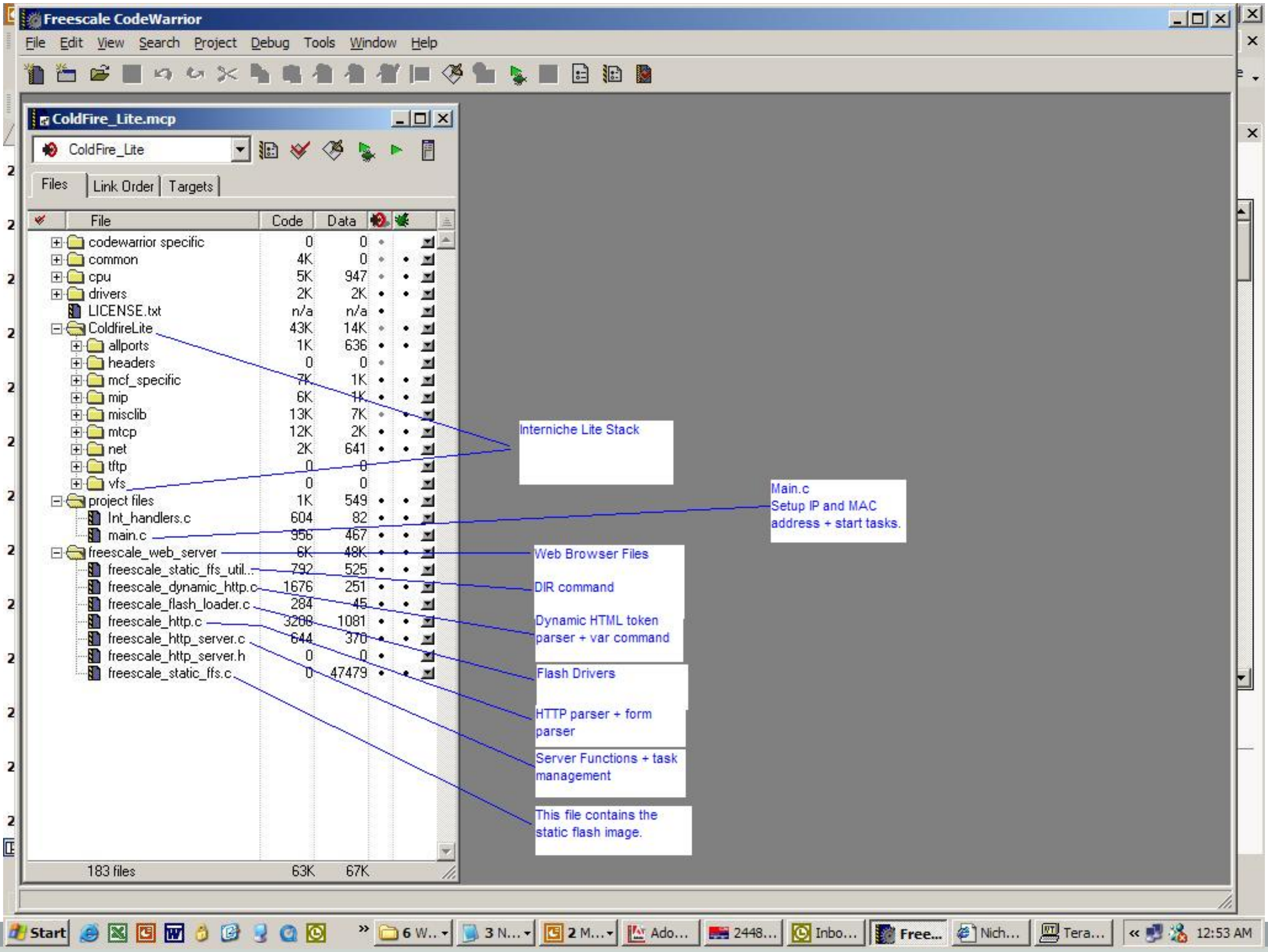
Name	Size	Type	Date Modified
ColdFire_Lite_Data		File Folder	5/11/2006 9:45 AM
ColdFire_Lite.mcp	99 KB	CodeWarrior Project	5/8/2006 7:42 PM
ColdFire_Lite.old.mcp	105 KB	CodeWarrior Project	4/8/2006 5:30 PM
m5223evb_pne.cfg	1 KB	CFG File	3/11/2006 2:23 PM

The ColdFireLite project


Freescale CodeWarrior
File Edit View Search Project Debug Tools Window Help
ColdFire_Lite.mcp
ColdFire_Lite
Files | Link Order | Targets

File	Code	Data
codewarrior specific	0	0
common	4K	0
cpu	5K	922
drivers	2K	2K
LICENSE.txt	n/a	n/a
ColdfireLite	50K	15K
project files	1K	533
freescale_web_server	6K	48K
183 files	71K	68K

Project Files



Freescale CodeWarrior — ColdFire_Lite.mcp — ColdFire_Lite

Files | Link Order | Targets

File	Code	Data		
codewarrior specific	0	0	·	
common	4K	0	·	●
cpu	5K	947	·	●
drivers	2K	2K	●	●
LICENSE.txt	n/a	n/a	●	
ColdfireLite	43K	14K	·	●
allports	1K	636	●	●
headers	0	0	·	
mcf_specific	7K	1K	●	●
mip	6K	1K	●	●
misclib	13K	7K	·	●
mtcp	12K	2K	●	●
net	2K	641	●	●
tftp	0	0		
vfs	0	0		
project files	1K	549	●	●
Int_handlers.c	604	82	●	●
main.c	956	467	●	●
freescale_web_server	6K	48K	●	●
freescale_static_ffs_util...	792	525	●	●
freescale_dynamic_http.c	1676	251	●	●
freescale_flash_loader.c	284	45	●	●
freescale_http.c	3208	1081	●	●
freescale_http_server.c	644	370	●	●
freescale_http_server.h	0	0	●	
freescale_static_ffs.c	0	47479	●	●
183 files	63K	67K		

Interniche Lite Stack

Main.c
Setup IP and MAC address + start tasks.

Web Browser Files

DIR command

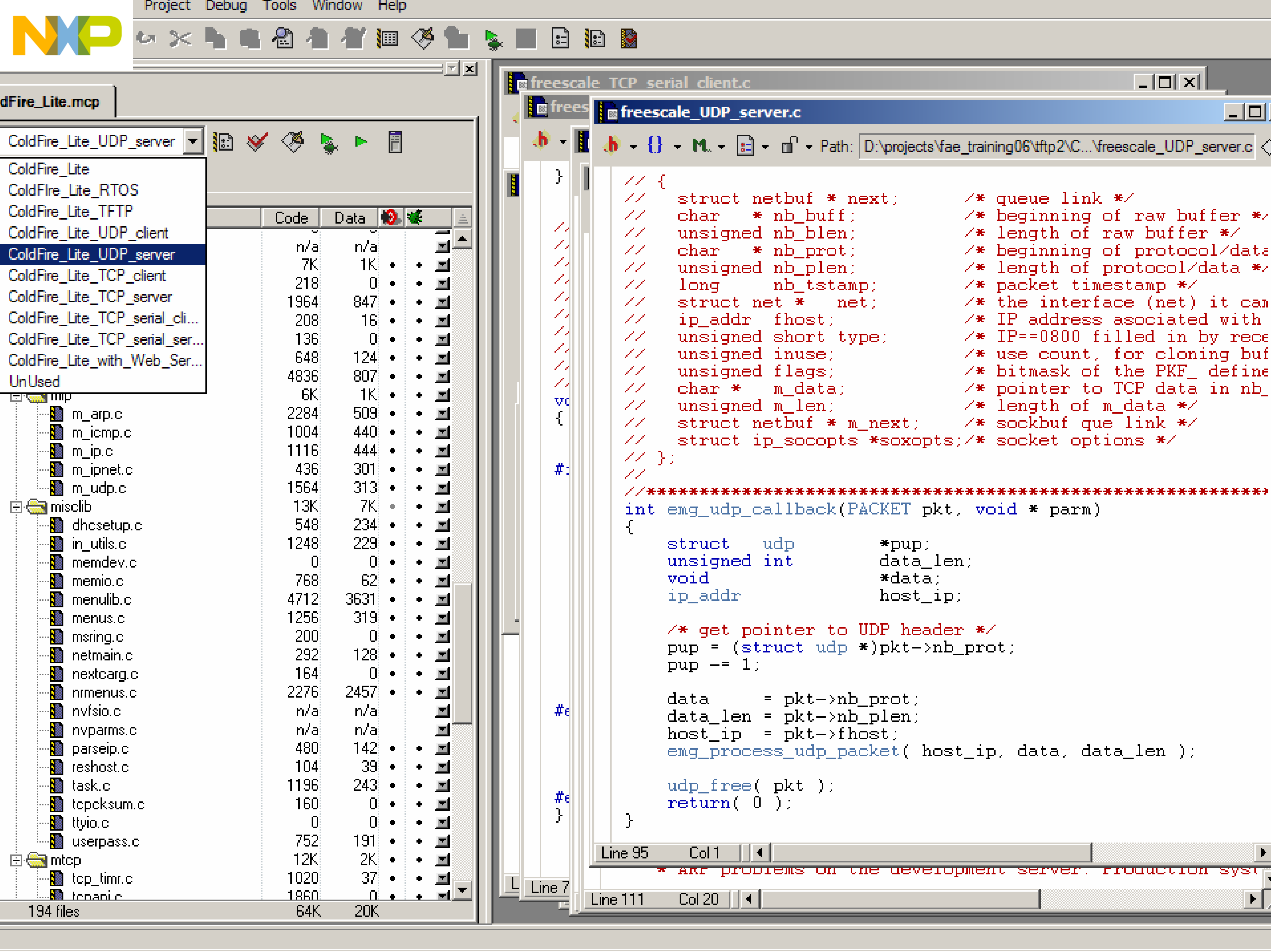
Dynamic HTML token parser + var command

Flash Drivers

HTTP parser + form parser

Server Functions + task management

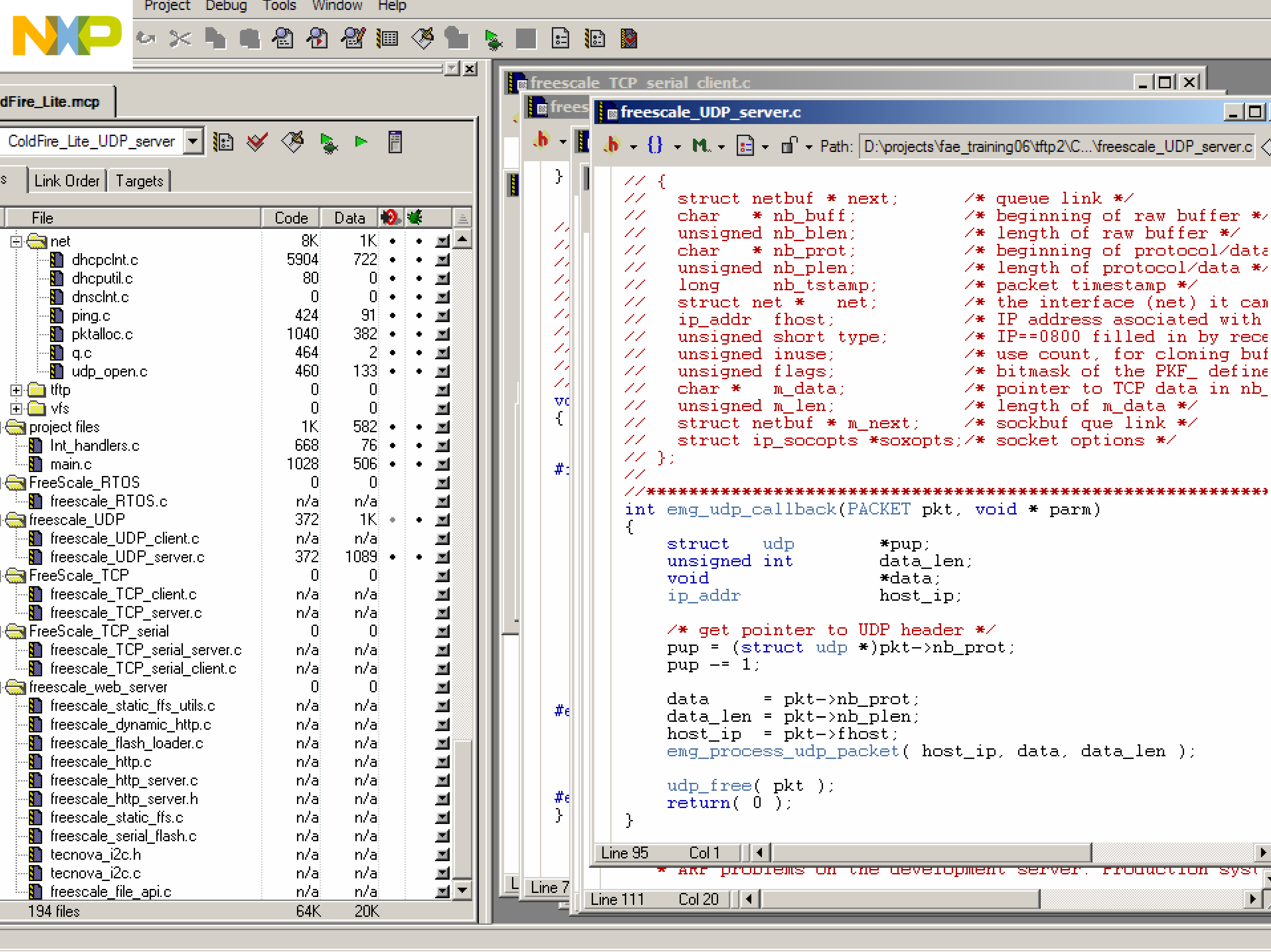
This file contains the static flash image.


Project Debug Tools Window Help
dFire_Lite.mcp
freescale_TCP_serial_client.c
frees
freescale_UDP_server.c
Path: D:\projects\fae_training06\tftp2\C...\freescale_UDP_server.c

```
ColdFire_Lite_UDP_server
ColdFire_Lite
ColdFire_Lite_RTOS
ColdFire_Lite_TFTP
ColdFire_Lite_UDP_client
ColdFire_Lite_UDP_server
ColdFire_Lite_TCP_client
ColdFire_Lite_TCP_server
ColdFire_Lite_TCP_serial_cli...
ColdFire_Lite_TCP_serial_ser...
ColdFire_Lite_with_Web_Ser...
UnUsed
```

	Code	Data
	n/a	n/a
	7K	1K
	218	0
	1964	847
	208	16
	136	0
	648	124
	4836	807
mip	6K	1K
m_arp.c	2284	509
m_icmp.c	1004	440
m_ip.c	1116	444
m_ipnet.c	436	301
m_udp.c	1564	313
misclib	13K	7K
dhcsetup.c	548	234
in_utils.c	1248	229
memdev.c	0	0
memio.c	768	62
menulib.c	4712	3631
menus.c	1256	319
msring.c	200	0
netmain.c	292	128
nextcarg.c	164	0
nrmenus.c	2276	2457
nvfsio.c	n/a	n/a
nvparms.c	n/a	n/a
parseip.c	480	142
reshost.c	104	39
task.c	1196	243
tcpcksum.c	160	0
ttyio.c	0	0
userpass.c	752	191
mtcp	12K	2K
tcp_timr.c	1020	37
tcpapi.c	1860	0
194 files	64K	20K

```
// {
//    struct netbuf * next;         /* queue link */
//    char    * nb_buff;            /* beginning of raw buffer */
//    unsigned nb_blen;             /* length of raw buffer */
//    char    * nb_prot;            /* beginning of protocol/data
//    unsigned nb_plen;             /* length of protocol/data */
//    long     nb_tstamp;           /* packet timestamp */
//    struct net *   net;           /* the interface (net) it cam
//    ip_addr  fhost;               /* IP address asociated with
//    unsigned short type;          /* IP==0800 filled in by rece
//    unsigned inuse;               /* use count, for cloning buf
//    unsigned flags;               /* bitmask of the PKF_ define
//    char *    m_data;             /* pointer to TCP data in nb_
//    unsigned m_len;               /* length of m_data */
//    struct netbuf * m_next;       /* sockbuf que link */
//    struct ip_socopts *soxopts;/* socket options */
// };
//
//*******************************************************************
int emg_udp_callback(PACKET pkt, void * parm)
{
    struct   udp          *pup;
    unsigned int          data_len;
    void                  *data;
    ip_addr               host_ip;

    /* get pointer to UDP header */
    pup = (struct udp *)pkt->nb_prot;
    pup -= 1;

    data     = pkt->nb_prot;
    data_len = pkt->nb_plen;
    host_ip  = pkt->fhost;
    emg_process_udp_packet( host_ip, data, data_len );

    udp_free( pkt );
    return( 0 );
}
```


Project Debug Tools Window Help
dFire_Lite.mcp
ColdFire_Lite_UDP_server
Link Order
Targets
freescale_TCP_serial_client.c
freescale_UDP_server.c
Path: D:\projects\fae_training06\tftp2\C...\freescale_UDP_server.c

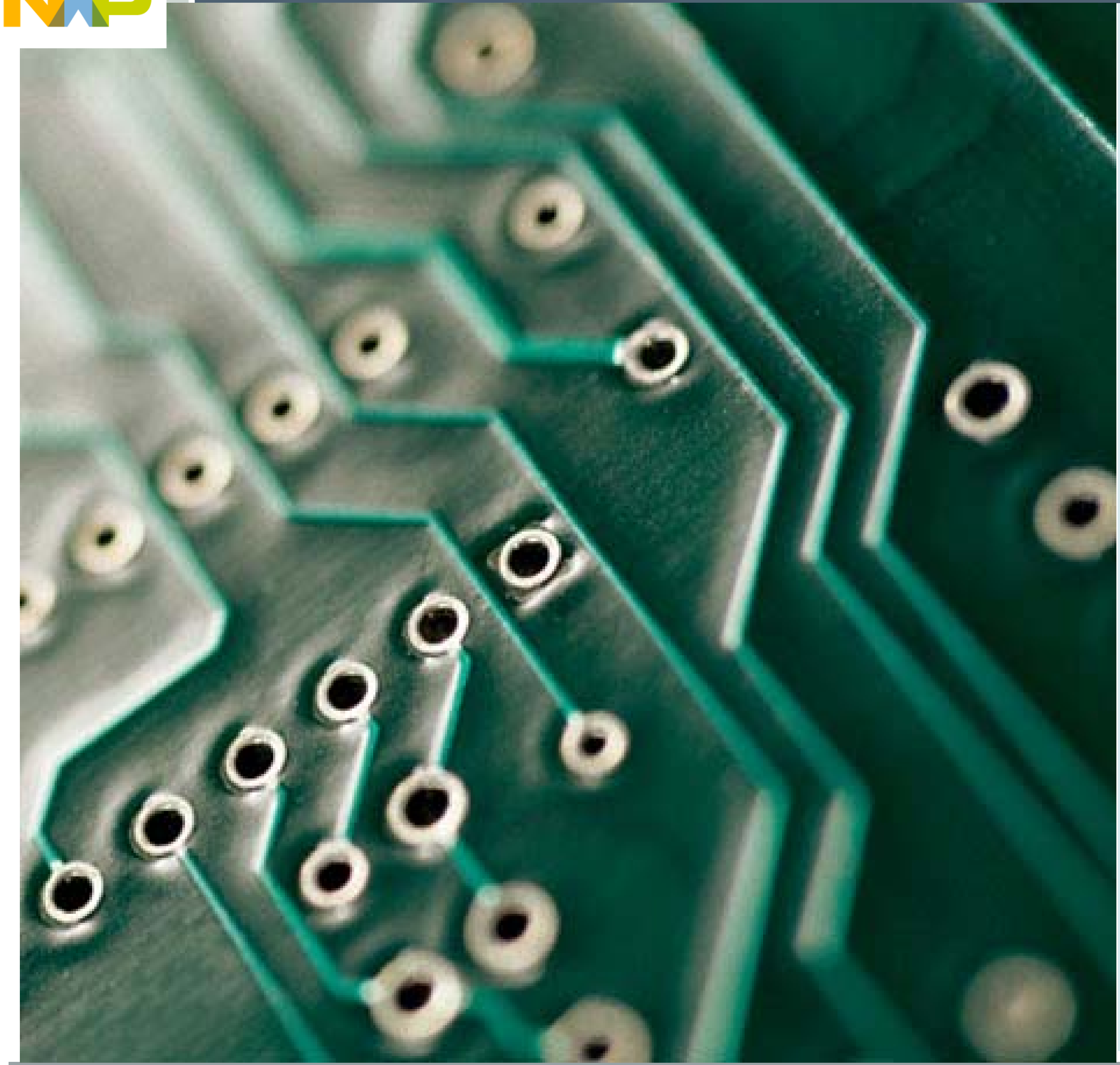
File	Code	Data		
net	8K	1K	•	•
dhcpclnt.c	5904	722	•	•
dhcputil.c	80	0	•	•
dnsclnt.c	0	0	•	•
ping.c	424	91	•	•
pktalloc.c	1040	382	•	•
q.c	464	2	•	•
udp_open.c	460	133	•	•
tftp	0	0		
vfs	0	0		
project files	1K	582	•	•
Int_handlers.c	668	76	•	•
main.c	1028	506	•	•
FreeScale_RTOS	0	0		
freescale_RTOS.c	n/a	n/a		
freescale_UDP	372	1K	•	•
freescale_UDP_client.c	n/a	n/a		
freescale_UDP_server.c	372	1089	•	•
FreeScale_TCP	0	0		
freescale_TCP_client.c	n/a	n/a		
freescale_TCP_server.c	n/a	n/a		
FreeScale_TCP_serial	0	0		
freescale_TCP_serial_server.c	n/a	n/a		
freescale_TCP_serial_client.c	n/a	n/a		
freescale_web_server	0	0		
freescale_static_ffs_utils.c	n/a	n/a		
freescale_dynamic_http.c	n/a	n/a		
freescale_flash_loader.c	n/a	n/a		
freescale_http.c	n/a	n/a		
freescale_http_server.c	n/a	n/a		
freescale_http_server.h	n/a	n/a		
freescale_static_ffs.c	n/a	n/a		
freescale_serial_flash.c	n/a	n/a		
tecnova_i2c.h	n/a	n/a		
tecnova_i2c.c	n/a	n/a		
freescale_file_api.c	n/a	n/a		
194 files	64K	20K		

```
// {
//    struct netbuf * next;        /* queue link */
//    char   * nb_buff;            /* beginning of raw buffer */
//    unsigned nb_blen;            /* length of raw buffer */
//    char   * nb_prot;            /* beginning of protocol/data
//    unsigned nb_plen;            /* length of protocol/data */
//    long     nb_tstamp;          /* packet timestamp */
//    struct net *   net;          /* the interface (net) it cam
//    ip_addr  fhost;              /* IP address asociated with
//    unsigned short type;         /* IP==0800 filled in by rece
//    unsigned inuse;              /* use count, for cloning buf
//    unsigned flags;              /* bitmask of the PKF_ define
//    char *   m_data;             /* pointer to TCP data in nb_
//    unsigned m_len;              /* length of m_data */
//    struct netbuf * m_next;      /* sockbuf que link */
//    struct ip_socopts *soxopts;/* socket options */
// };
//
//***************************************************************
int emg_udp_callback(PACKET pkt, void * parm)
{
    struct   udp          *pup;
    unsigned int          data_len;
    void                  *data;
    ip_addr               host_ip;

    /* get pointer to UDP header */
    pup = (struct udp *)pkt->nb_prot;
    pup -= 1;

    data     = pkt->nb_prot;
    data_len = pkt->nb_plen;
    host_ip  = pkt->fhost;
    emg_process_udp_packet( host_ip, data, data_len );

    udp_free( pkt );
    return( 0 );
}
```

Line 95 Col 1

* ARP problems on the development server. Production syst

Line 111 Col 20



LAB: Flashing the board with ColdFire_Lite

Flashing and Booting the board

Connect the board via USB and serial to the PC.

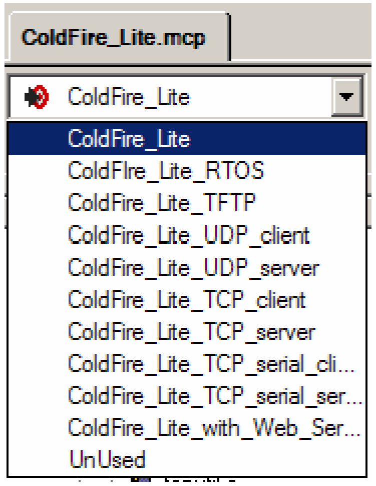
Select the ColdFire_Lite project

Flash the ColdFire_Lite project

Setup Hyperterminal for

- 115200, 8, n, 1

Run the project

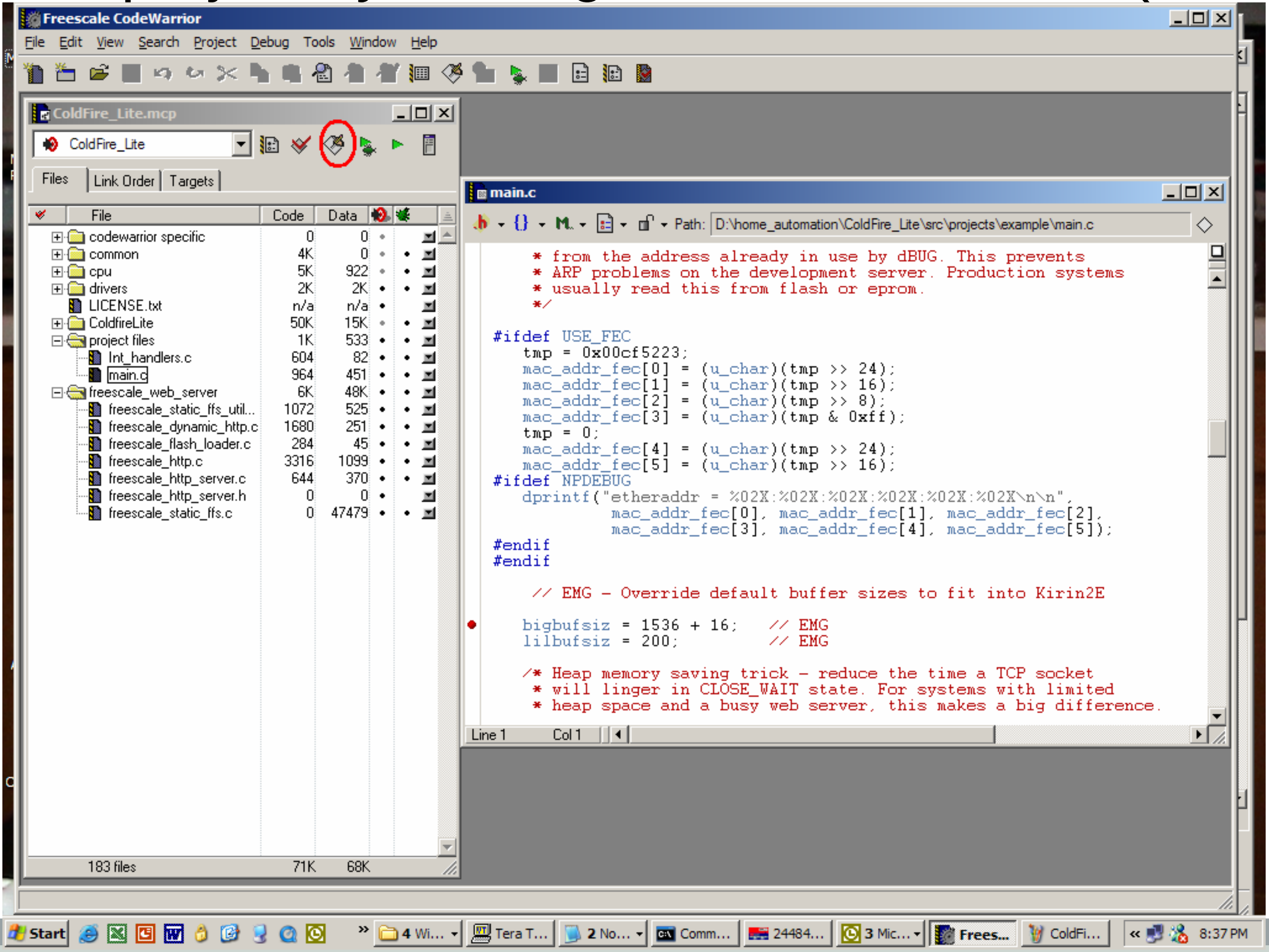


ColdFire_Lite.mcp

ColdFire_Lite

ColdFire_Lite
ColdFIre_Lite_RTOS
ColdFire_Lite_TFTP
ColdFire_Lite_UDP_client
ColdFire_Lite_UDP_server
ColdFire_Lite_TCP_client
ColdFire_Lite_TCP_server
ColdFire_Lite_TCP_serial_cli...
ColdFire_Lite_TCP_serial_ser...
ColdFire_Lite_with_Web_Ser...
UnUsed

Build the project by clicking on the MAKE icon (circled in **RED**)



File	Code	Data		
codewarrior specific	0	0	•	
common	4K	0	•	•
cpu	5K	922	•	•
drivers	2K	2K	•	•
LICENSE.txt	n/a	n/a	•	
ColdfireLite	50K	15K	•	•
project files	1K	533	•	•
Int_handlers.c	604	82	•	•
main.c	964	451	•	•
freescale_web_server	6K	48K	•	•
freescale_static_ffs_util...	1072	525	•	•
freescale_dynamic_http.c	1680	251	•	•
freescale_flash_loader.c	284	45	•	•
freescale_http.c	3316	1099	•	•
freescale_http_server.c	644	370	•	•
freescale_http_server.h	0	0	•	
freescale_static_ffs.c	0	47479	•	•
183 files	71K	68K		

Path: D:\home_automation\ColdFire_Lite\src\projects\example\main.c

```
     * from the address already in use by dBUG. This prevents
     * ARP problems on the development server. Production systems
     * usually read this from flash or eprom.
     */

#ifdef USE_FEC
   tmp = 0x00cf5223;
   mac_addr_fec[0] = (u_char)(tmp >> 24);
   mac_addr_fec[1] = (u_char)(tmp >> 16);
   mac_addr_fec[2] = (u_char)(tmp >> 8);
   mac_addr_fec[3] = (u_char)(tmp & 0xff);
   tmp = 0;
   mac_addr_fec[4] = (u_char)(tmp >> 24);
   mac_addr_fec[5] = (u_char)(tmp >> 16);
#ifdef NPDEBUG
   dprintf("etheraddr = %02X:%02X:%02X:%02X:%02X:%02X\n\n",
            mac_addr_fec[0], mac_addr_fec[1], mac_addr_fec[2],
            mac_addr_fec[3], mac_addr_fec[4], mac_addr_fec[5]);
#endif
#endif

    // EMG - Override default buffer sizes to fit into Kirin2E

   bigbufsiz = 1536 + 16;   // EMG
   lilbufsiz = 200;         // EMG

   /* Heap memory saving trick - reduce the time a TCP socket
    * will linger in CLOSE_WAIT state. For systems with limited
    * heap space and a busy web server, this makes a big difference.
```

Line 1   Col 1

Flash Programming Caution

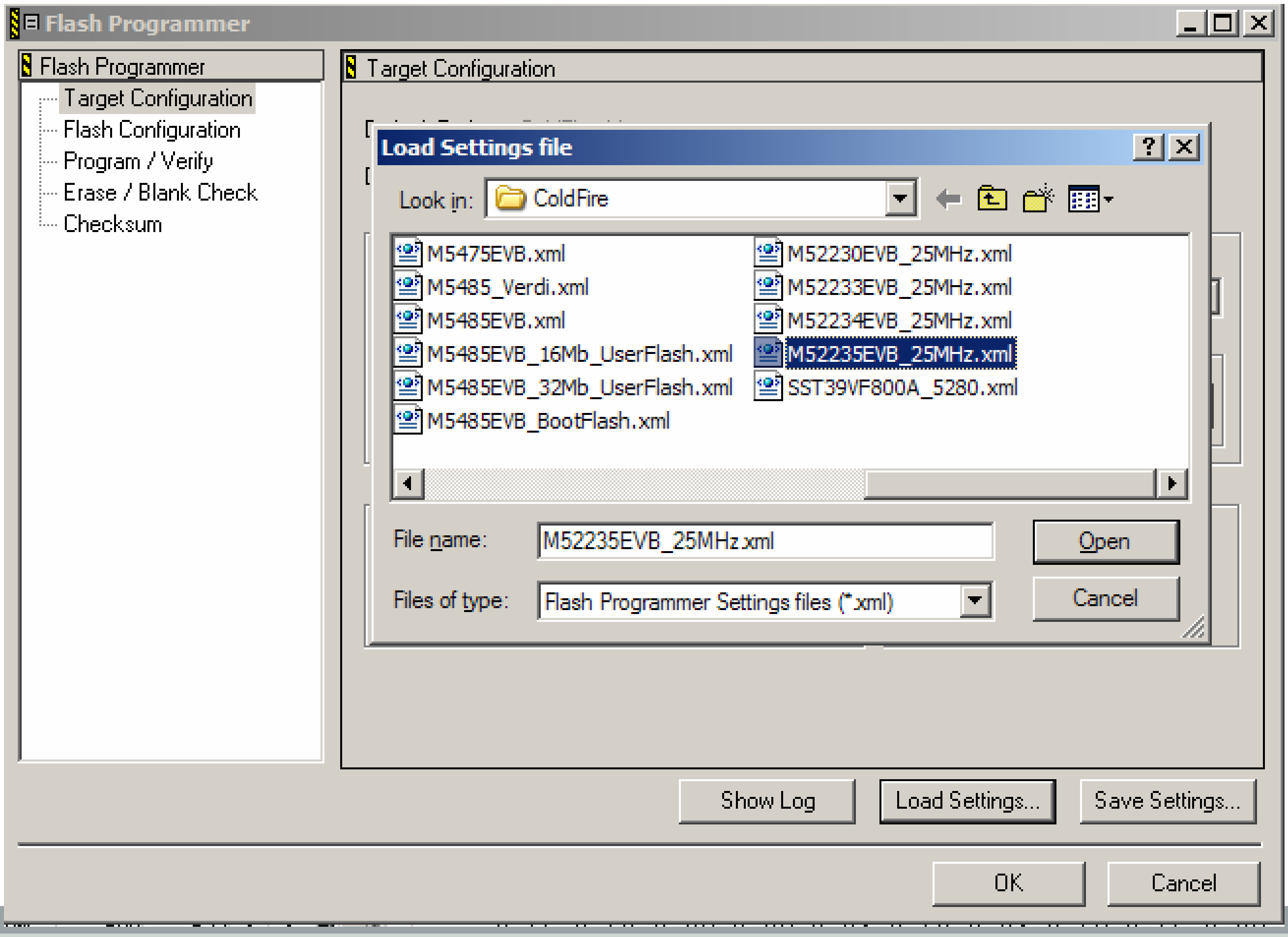
CodeWarrior® for ColdFire **DOES NOT** behave like the HC08 and HC12 tools when downloading code to internal FLASH.

Code **MUST** be downloaded by the Flash programmer to internal Flash of the MCF52235 as described in the previous slides.

Once code is programmed in Flash it can be debugged using the procedure in the following slides.

Selecting the XML File

Select the M5223EVB-25MHZ xml file.



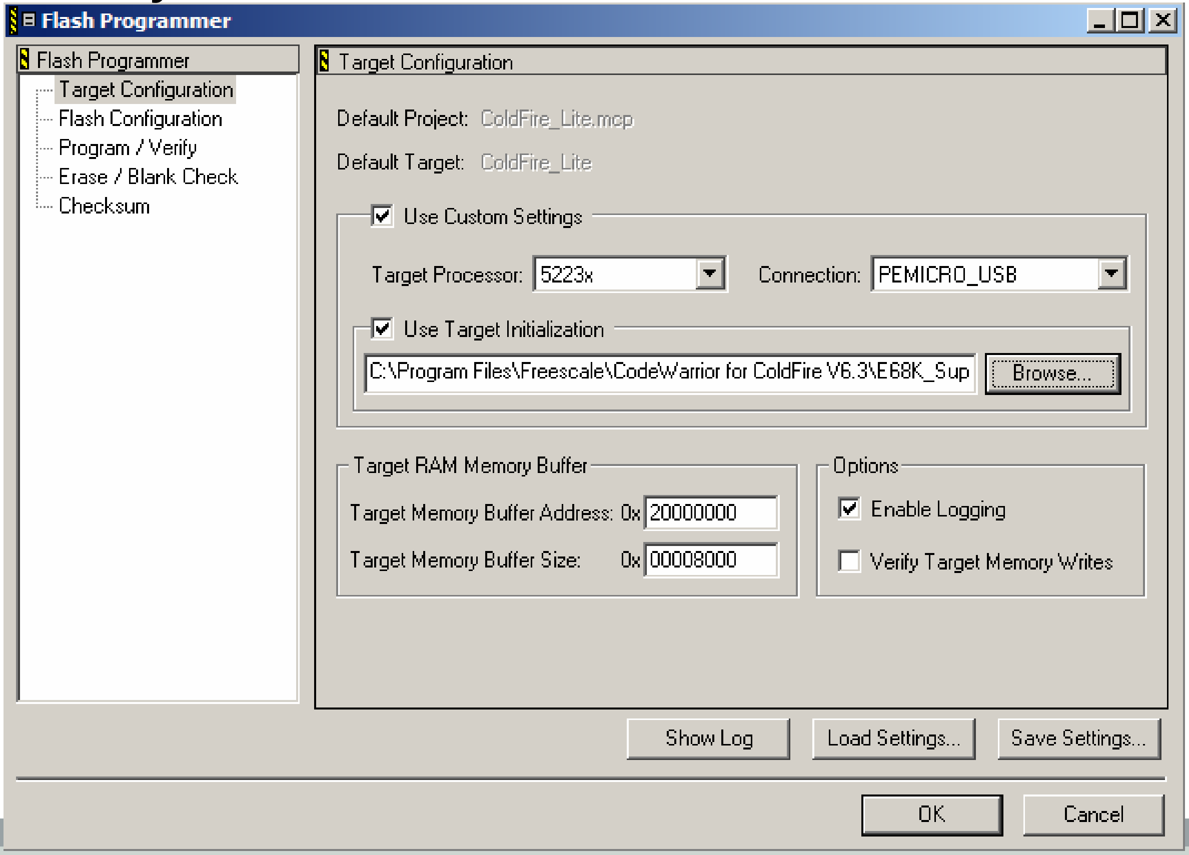
M5475EVB.xml
M5485_Verdi.xml
M5485EVB.xml
M5485EVB_16Mb_UserFlash.xml
M5485EVB_32Mb_UserFlash.xml
M5485EVB_BootFlash.xml
M52230EVB_25MHz.xml
M52233EVB_25MHz.xml
M52234EVB_25MHz.xml
M52235EVB_25MHz.xml
SST39VF800A_5280.xml

File name: M52235EVB_25MHz.xml

Files of type: Flash Programmer Settings files (*.xml)

Flash Programmer Screen

After Loading the XML file, the Flash Programmer will show following screen. Note the Target Processor, and RAM memory buffers are setup automatically from the XML file.



Default Project: ColdFire_Lite.mcp

Default Target: ColdFire_Lite

[✓] Use Custom Settings

Target Processor: 5223x

Connection: PEMICRO_USB

[✓] Use Target Initialization

C:\Program Files\Freescale\CodeWarrior for ColdFire V6.3\E68K_Sup

Target RAM Memory Buffer

Target Memory Buffer Address: 0x 20000000

Target Memory Buffer Size: 0x 00008000

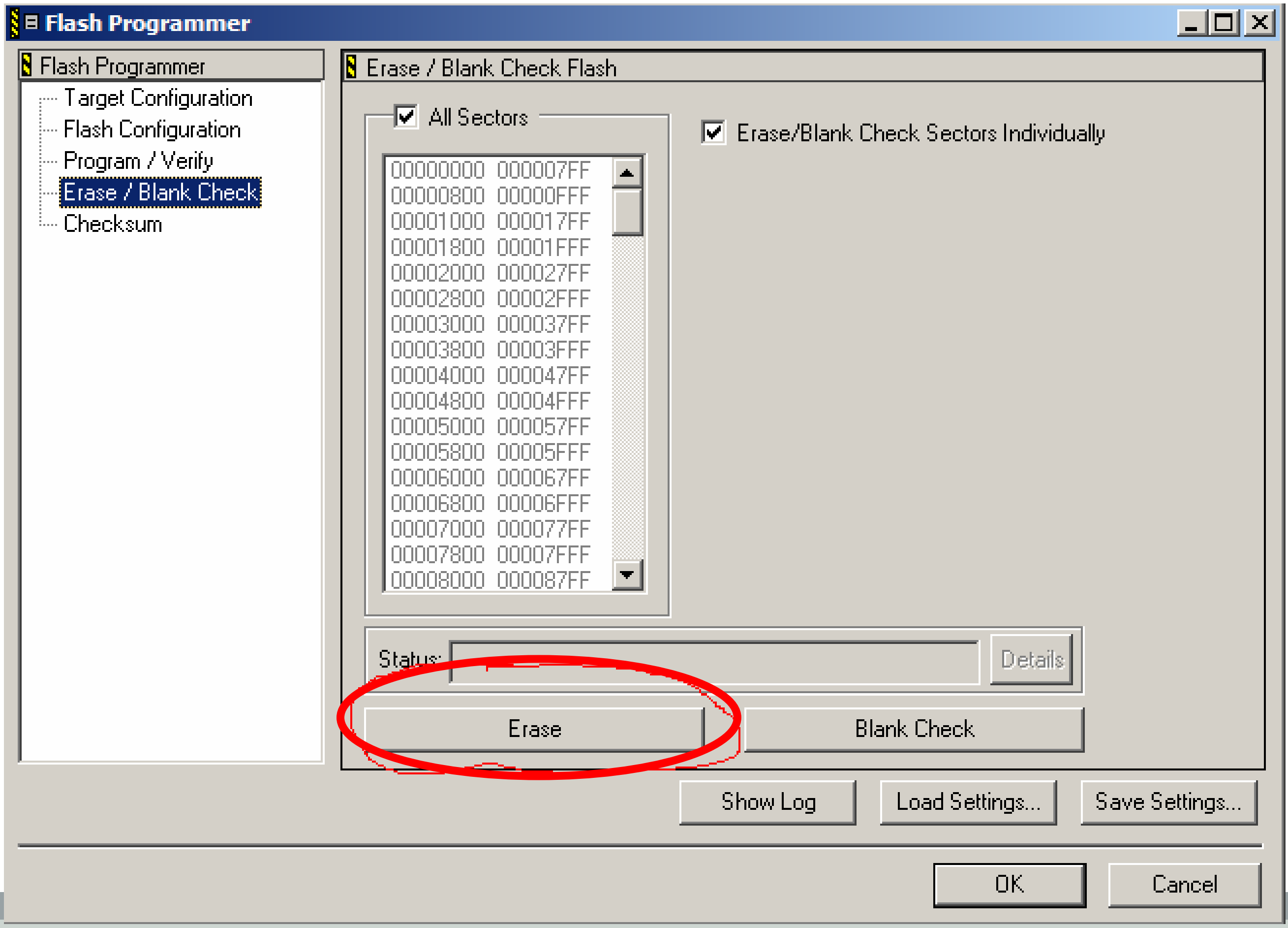
Options

[✓] Enable Logging

[ ] Verify Target Memory Writes

Erase Flash

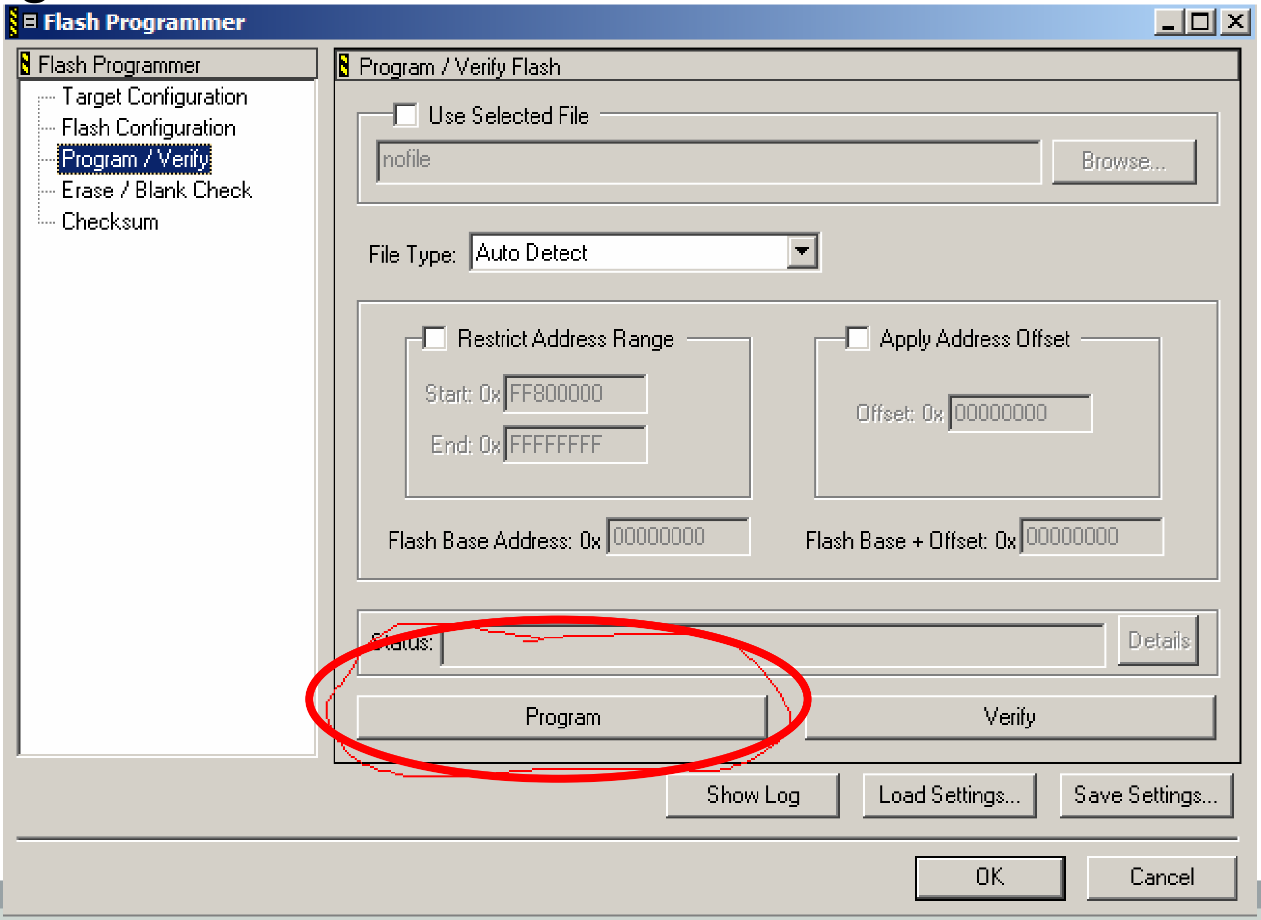
Erase the Flash by selecting Erase/Blank Check, and clicking the Erase button. Watch the Status window for errors.



```
00000000  000007FF
00000800  00000FFF
00001000  000017FF
00001800  00001FFF
00002000  000027FF
00002800  00002FFF
00003000  000037FF
00003800  00003FFF
00004000  000047FF
00004800  00004FFF
00005000  000057FF
00005800  00005FFF
00006000  000067FF
00006800  00006FFF
00007000  000077FF
00007800  00007FFF
00008000  000087FF
```

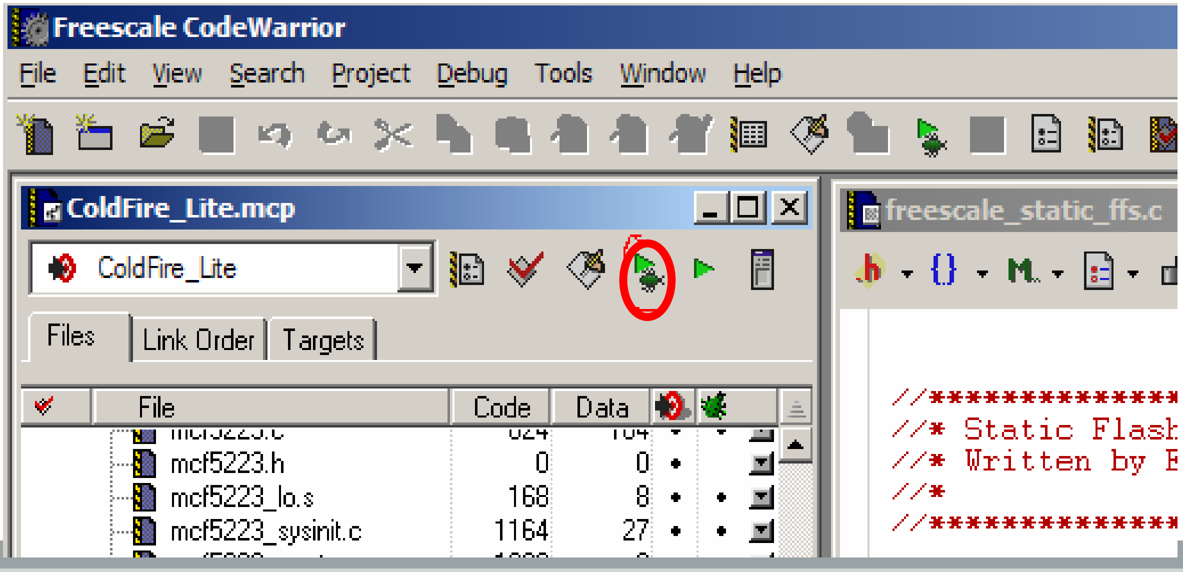
Programming the Flash

After the Erase is Complete, go to the Program/Verify window and click on the Program button.



RUN

Click on the Run icon, circled in RED below. This will execute the code in flash. If you have an external power supply, you could also disconnect the USB from the board and hit reset.



	File	Code	Data		
	mcf5223.h	0	0	•	
	mcf5223_lo.s	168	8	•	•
	mcf5223_sysinit.c	1164	27	•	•

```
//****************
//* Static Flash
//* Written by E
//*
//****************
```

Connecting the Serial Port

Connect the serial port on the demo board to the PC. Then open hyperterminal and configure for 115Kbaud, 8, n, 1, no flow control. Hit enter until you see the ‘INET>’ prompt then type ‘tkstat’.

Serial output at boot (115200, 8, n, 1)

```
Running ColdFire TCP/IP-Lite stack

Copyright 2006 by Freescale Semiconductor Inc.
Use of this software is controlled by the agreement
found in the project LICENSE.H file.
Built on Sep 27 2006 19:25:56

Heap size = 27136 bytes

IP Address = C0A80163
Gateway    = C0A80101
Mask       = FFFFFF00
etheraddr = 00:BA:DB:AD:01:02

Starting ints.
Calling netmain()...
InterNiche ColdFireLite TCP/IP for Coldfire, v3.0

Copyright 1997-2006 by InterNiche Technologies. All rights reserved.
Preparing device for networking
Ethernet started, Iface: 0, IP: 192.168.1.99
IP address of  : 192.168.1.99
INET>
INET>
```

The ColdFire_Lite target

The ColdFire_Lite target is the barbones Interniche stack only target.

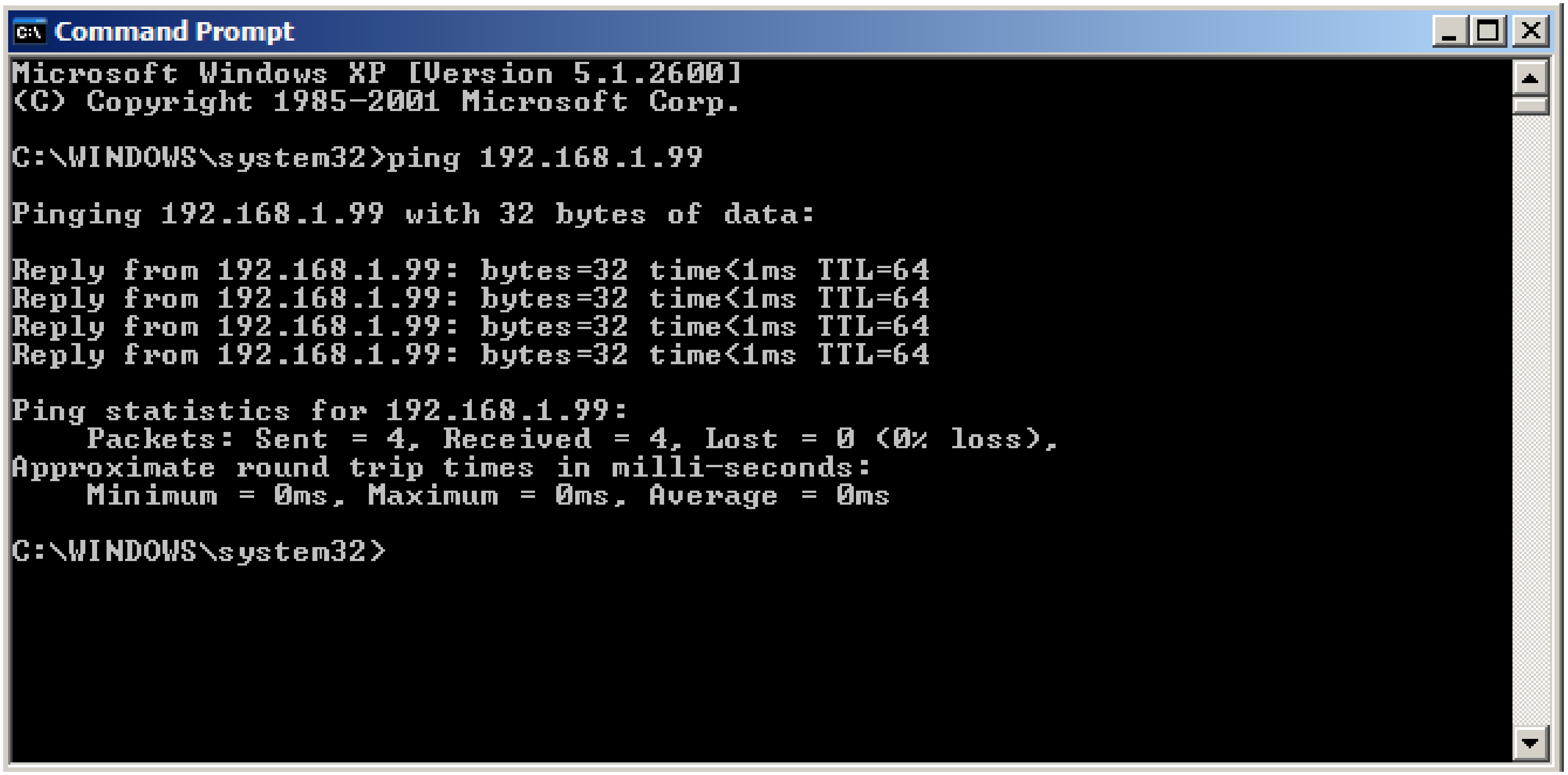
Use this project for customers who are looking for a clean port of the Interniche stack to the MCF5223x.

The only features available with this target is the serial console and ping.

Ping the ColdFire from the PC

Open a DOS window

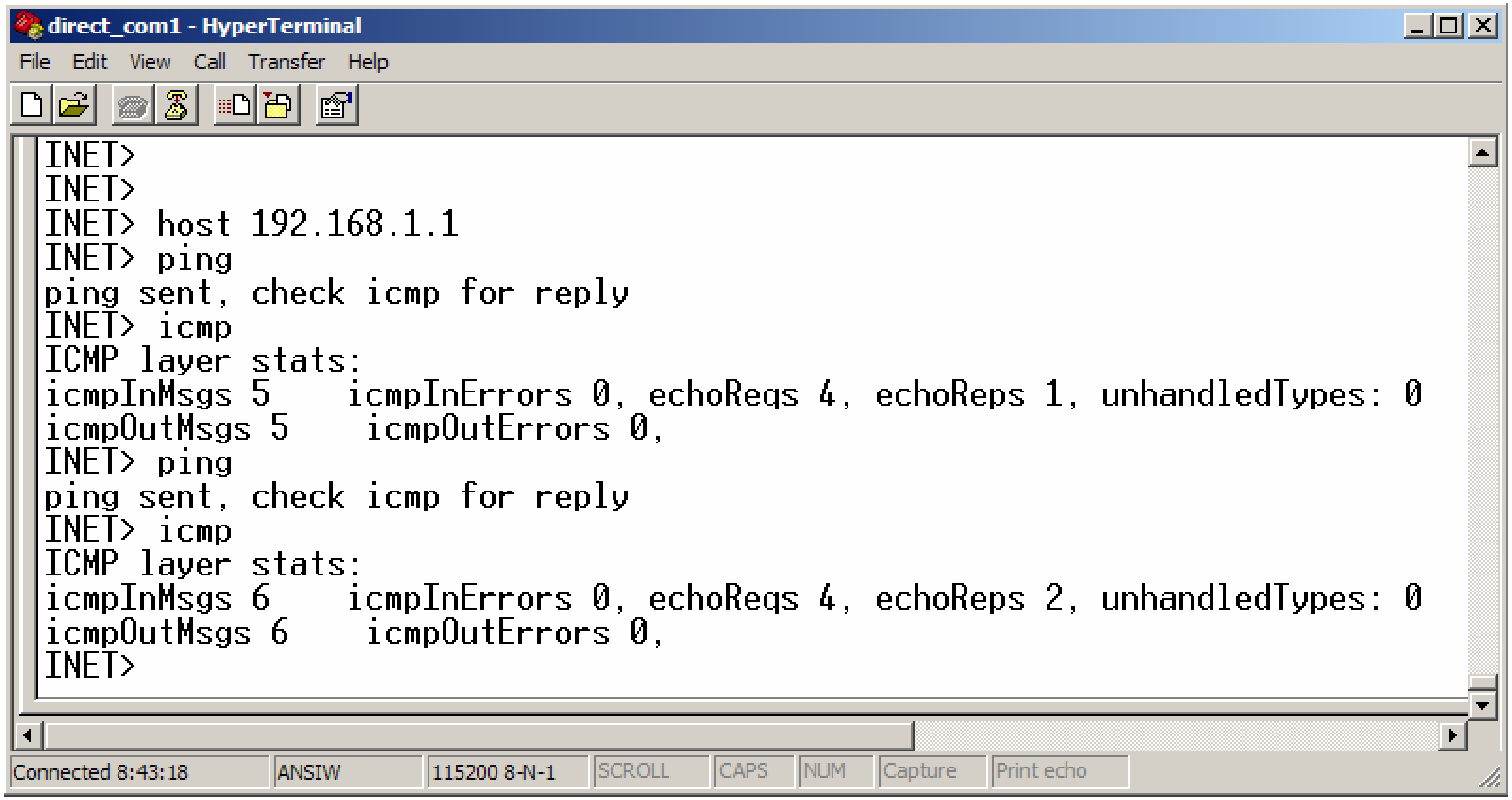
Type: ping 192.168.1.99 (the ColdFire default IP address)



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ping 192.168.1.99

Pinging 192.168.1.99 with 32 bytes of data:

Reply from 192.168.1.99: bytes=32 time<1ms TTL=64
Reply from 192.168.1.99: bytes=32 time<1ms TTL=64
Reply from 192.168.1.99: bytes=32 time<1ms TTL=64
Reply from 192.168.1.99: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.99:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\WINDOWS\system32>
```

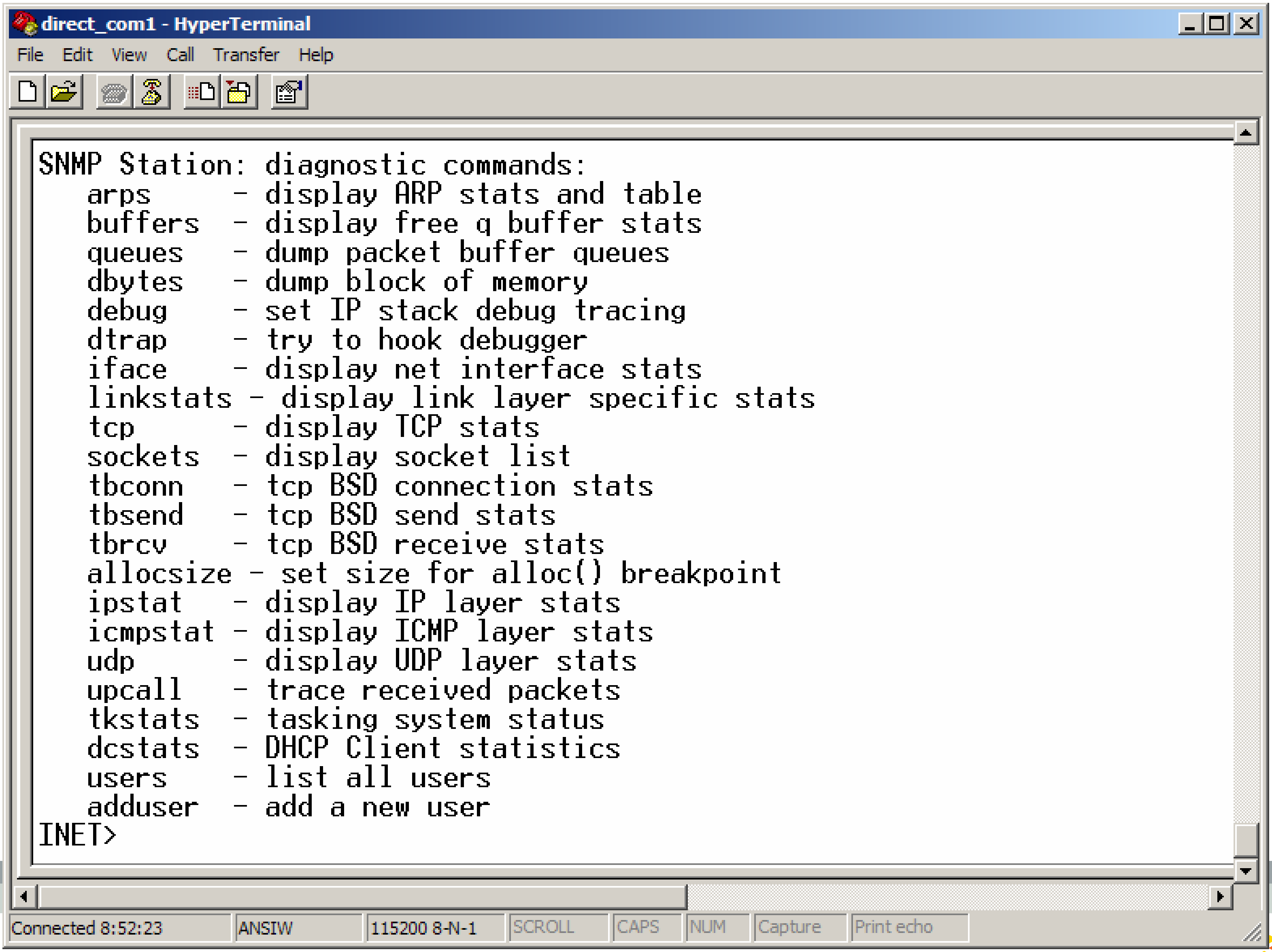
Ping the PC from the ColdFire

At the INET prompt type:



```
INET>
INET>
INET> host 192.168.1.1
INET> ping
ping sent, check icmp for reply
INET> icmp
ICMP layer stats:
icmpInMsgs 5    icmpInErrors 0, echoReqs 4, echoReps 1, unhandledTypes: 0
icmpOutMsgs 5    icmpOutErrors 0,
INET> ping
ping sent, check icmp for reply
INET> icmp
ICMP layer stats:
icmpInMsgs 6    icmpInErrors 0, echoReqs 4, echoReps 2, unhandledTypes: 0
icmpOutMsgs 6    icmpOutErrors 0,
INET>
```

Type: help diag



```
SNMP Station: diagnostic commands:
   arps      - display ARP stats and table
   buffers   - display free q buffer stats
   queues    - dump packet buffer queues
   dbytes    - dump block of memory
   debug     - set IP stack debug tracing
   dtrap     - try to hook debugger
   iface     - display net interface stats
   linkstats - display link layer specific stats
   tcp       - display TCP stats
   sockets   - display socket list
   tbconn    - tcp BSD connection stats
   tbsend    - tcp BSD send stats
   tbrcv     - tcp BSD receive stats
   allocsize - set size for alloc() breakpoint
   ipstat    - display IP layer stats
   icmpstat  - display ICMP layer stats
   udp       - display UDP layer stats
   upcall    - trace received packets
   tkstats   - tasking system status
   dcstats   - DHCP Client statistics
   users     - list all users
   adduser   - add a new user
INET>
```

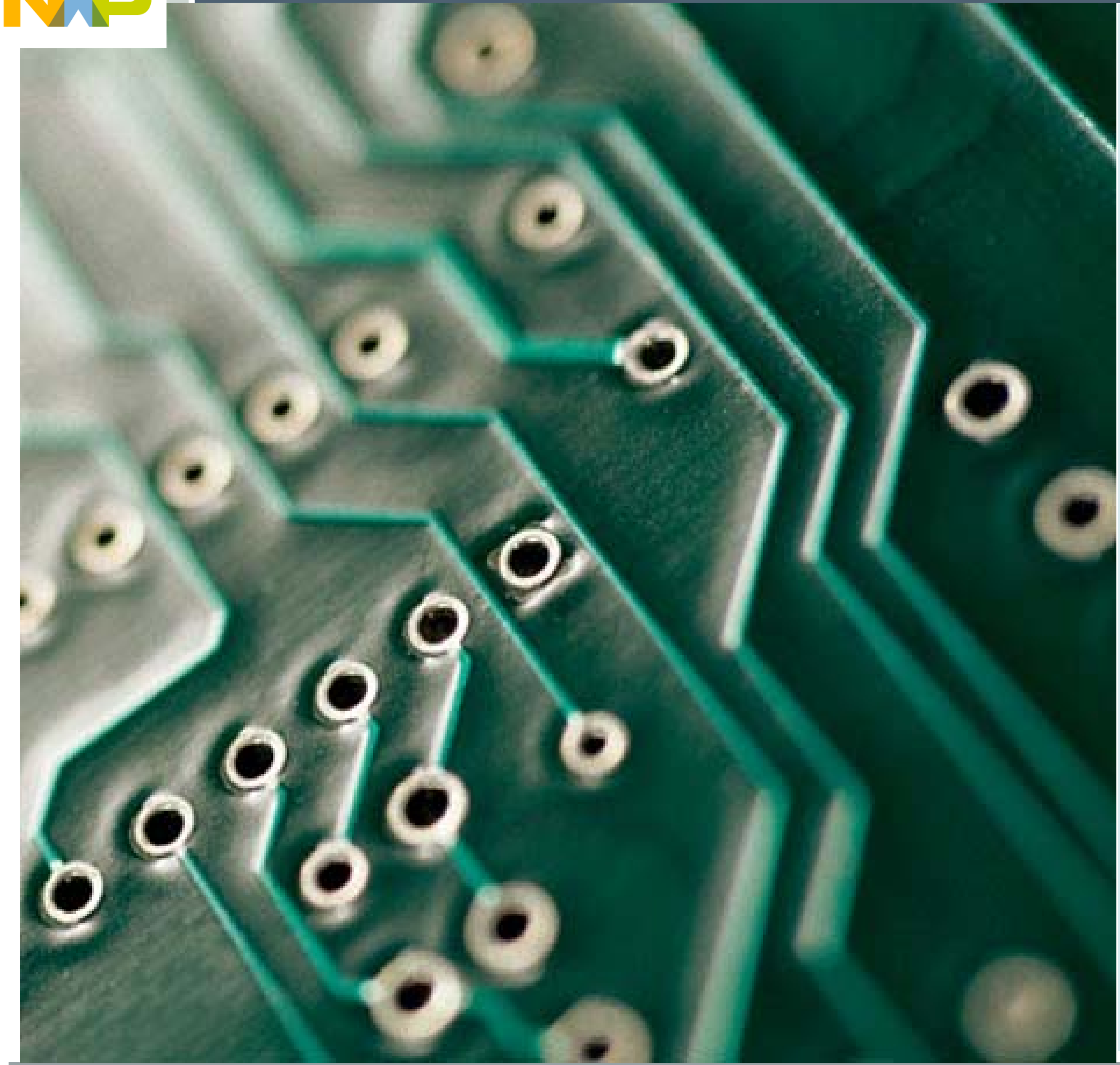
To view the running tasks

```
INET> tkstats
tasking status:task wakeups: D
   name              state        stack   used    wakes
console              running      2048    556     153035
clock tick           sleeping     2048    204     117764
Main                 blocked      2048    456     202
INET>
```

console = The serial console.

clock tick = RTOS/Stack timers.

Main = “Inet main” = The Stack task.



ColdFire Boot Up Sequence

ColdFire_Lite Boot

In file MCF5223_VECTORS.S

```
/*
 * Exception Vector Table
 */
VECTOR_TABLE:
_VECTOR_TABLE:
INITSP:              .long      ___SP_INIT/* Initial SP                      */
INITPC:              .long      start              /* Initial PC                      */
……………………………………………………………………………………………………..
start:
          move.w     #0x2700,sr
          jmp        _asm_startmeup
```

In file MCF5223_lo.s

```
_asm_startmeup:
…………
          jsr                  mcf5223_init         // mcf5223_sysinit.c
          jsr                  cpu_startup // mcf5223.c
…………
          /* Jump to the main process */
          jsr                  main
```

ColdFire_Lite Boot main.c

```
 /* hardcode FEC IP address for now. We set it in netstatic, and
    * Ip startup code will initialize net[] from it.
    */
#if 1  // EMG 192.168.1.99
   netstatic[0].n_ipaddr = (0xC0A80163);
   netstatic[0].n_defgw  = (0x00000000);
   netstatic[0].snmask   = (0xffffff00);
#else  //jpw  192.168.2.3
   netstatic[0].n_ipaddr = (0xC0A80203);
   netstatic[0].n_defgw  = (0xC0A80201);
   netstatic[0].snmask   = (0xffffff00);
#endif

   netstatic[0].mib.ifDescr = (u_char *)"Fast Ethernet Controller";

   /* We set the station's Ethernet physical (MAC) address
    * from the address already in use by dBUG. This prevents
    * ARP problems on the development server. Production systems
    * usually read this from flash or eprom.
    */

#ifdef USE_FEC
   tmp = 0x00cf5223;
   mac_addr_fec[0] = (u_char)(tmp >> 24);
   mac_addr_fec[1] = (u_char)(tmp >> 16);
   mac_addr_fec[2] = (u_char)(tmp >> 8);
   mac_addr_fec[3] = (u_char)(tmp & 0xff);
   tmp = 0;
   mac_addr_fec[4] = (u_char)(tmp >> 24);
   mac_addr_fec[5] = (u_char)(tmp >> 16);
#ifdef NPDEBUG
   dprintf("etheraddr = %02X:%02X:%02X:%02X:%02X:%02X\n\n",
           mac_addr_fec[0], mac_addr_fec[1], mac_addr_fec[2],
           mac_addr_fec[3], mac_addr_fec[4], mac_addr_fec[5]);
#endif
#endif
```

Main.c

```
                // EMG - Override default buffer sizes to fit into Kirin2E

   bigbufsiz = 1536 + 16;                                  // EMG
   lilbufsiz = 200;                                        // EMG

   /* Heap memory saving trick - reduce the time a TCP socket
    * will linger in CLOSE_WAIT state. For systems with limited
    * heap space and a busy web server, this makes a big difference.
    */
   // EMG was 5    4/5/06 set to 1
   TCPTV_MSL = 1;    /* set low max seg lifetime default */

#ifdef NPDEBUG
   printf("Starting ints.\n");
#endif

//   mcf5xxx_irq_enable();          /* Let the interrupts fly... */
   iniche_net_ready = TRUE;

                while( !uart_flush(0) ){};

#ifdef NPDEBUG
   printf("Calling netmain()...\n");
#endif
   netmain();     /* Start and run net tasks, no return. */
   USE_ARG(err);
   return 0;
}
```

ColdFire_Lite Boot netmain()

Main() calls netmain() in netmain.c which starts all the tasks.

```
int
netmain(void)
{
   int   i;
   int   e;

   iniche_net_ready = FALSE;

   e = prep_modules(); --------------------------------------------- in allports.c

   /* Create the threads for net, timer, and apps */
   for (i = 0; i < num_net_tasks; i++)
   {
      e = TK_NEWTASK(&nettasks[i]); -------------------------- Walk through the nettask[] array in netmain.c, “Inet main” MUST be first.
      if (e != 0)
      {
         dprintf("task create error\n");
         panic("netmain");
         return -1;  /* compiler warnings */
      }
   }

   e = create_apptasks(); ----------------------------------------- Starts the FreeScale task, and console task
   if (e != 0)
   {
      dprintf("task create error\n");
      panic("netmain");
      return -1;  /* compiler warnings */
   }

   uart_yield = 1;

                          // MAIN_TASK_IS_NET
   tk_netmain(TK_NETMAINPARM); -------------------------- Starts tk_netmain in netmain.c.  This is the main network task.
   panic("net task return"); ---------------------------------------- The task never returns.
   return -1;
}
```

Network tasks

```
struct inet_taskinfo nettasks[]  =  {
#ifndef NO_INET_STACK
   {
      &to_netmain,   /* netmain should always be first in this array */
      "Inet main",
      tk_netmain,
      NET_PRIORITY,
      NET_STACK_SIZE,
   },
#endif   /* NO_INET_STACK */
#ifndef NO_INET_TICK

   {
      &to_nettick,
      "clock tick",
      tk_nettick,
      NET_PRIORITY,
      CLOCK_STACK_SIZE,
   },
#endif   /* NO_INET_TICK */
```

ColdFire_Lite Boot, the network task

In allports.c

```
/* FUNCTION: tk_netmain()
 *
 * Main thread for starting the net. After startup, it settles into
 * a loop handling received packets. This loop sleeps until a packet
 * has been queued in rcvdq; at which time it should be awakend by the
 * driver which queued the packet.
 *
 * PARAM1: n/a
 *
 * RETURNS: n/a
 */

#ifndef NO_INET_STACK
TK_ENTRY(tk_netmain)
{
   netmain_init(); /* initialize all modules */

   iniche_net_ready = TRUE;    /* let the other threads spin */

   for (;;)
   {
      TK_NETRX_BLOCK();
      netmain_wakes++;  /* count wakeups */

      /* see if there's newly received network packets */
      if (rcvdq.q_len)
         pktdemux();
      /* do not kill packet demux on net_system_exit. It may be
       * vital to a clean shutdown
       */
   }
   USE_ARG(parm);  /* TK_ENTRY macro defines tk_netmain with 1 arg parm */
   TK_RETURN_UNREACHABLE();
}
```

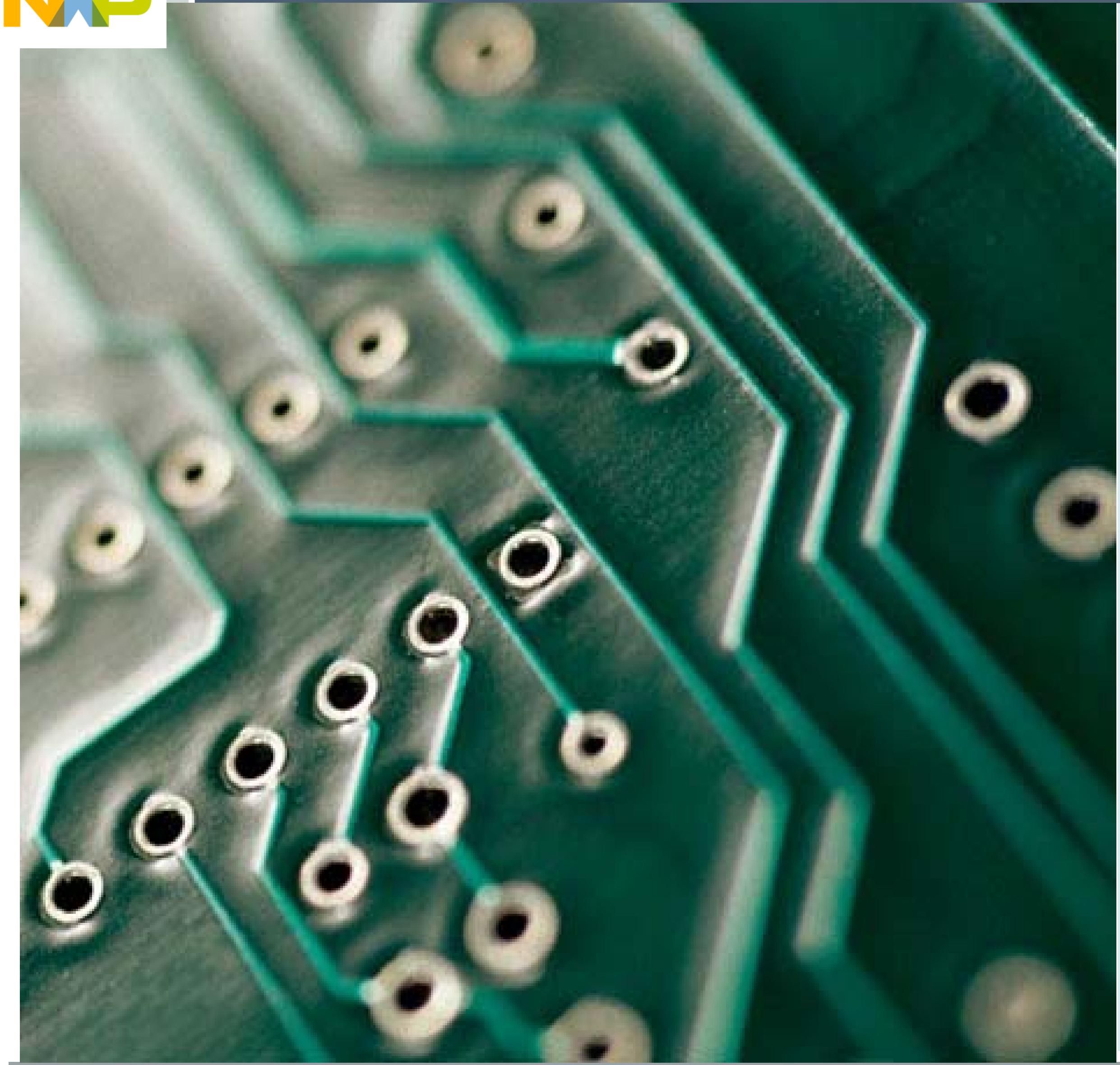
FreeScale Task

The function create_apptasks() in tk_misc.c creates the FreeScale task used in all the future labs.

The FreeScale task then starts any additional tasks required for that particular folder.

In tk_misc.c

```
int
create_apptasks(void)
{
int e = 0;

#ifndef TFTP_PROJECT
// EMG
             create_freescale_task();
#endif

#ifdef TK_STDIN_DEVICE
  e = TK_NEWTASK(&keyboardtask);
  if (e != 0)
  {
    dprintf("keyboardtask create error\n");
    panic("create_apptasks");
    return -1;  /* compiler warnings */
  }
#endif

return 0;
}
```



The InterNiche RTOS

NicheTASK

The Interniche stack also contains a simple RTOS.

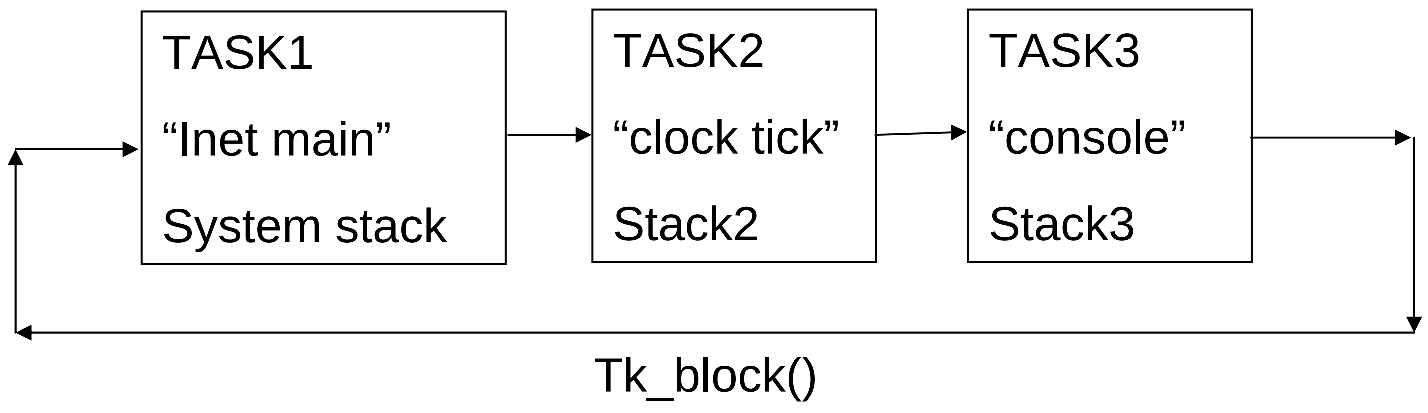
- Non-preemptive – requires that a task give up control to the next task.
- Each task has its own stack (not superloop).
- A task is either sleeping based on time or a event.
- If a task is not sleeping, it is ready to run.

- You can add your own task via the tk_new() function.
- Your task MUST sleep to give up control to the next task in the list.

- There are no priorities, when task 1 gives up control, the RTOS tries to run task 2, and so on.

- Task 1 uses the system task, and MUST be the network task.

RTOS



tk_block() switches from one task to the next task in the ring.

tk_new() adds a new task into the ring.

Tk_kill() removes the stack from the ring.

If a task is “sleeping”, it is skipped in the ring.

RTOS Functions

```
// entry points to tasking system
task *  tk_init(stack_t * base, int st_size);          // Init the RTOS
task *  tk_new(task*, int(*)(int), int, char*, int);   // add a new task to the list
void    tk_block(void);                                // switch to next runnable task
void    tk_exit(void);                                 // kill & delete current task
void    tk_kill(task * tk_to_die);                     // mark any task for death
void    tk_wake(task * tk);                            // mark a task to run
void    tk_sleep(long ticks);                          // sleep for number of ticks
void    tk_ev_block(void * event);                     // block until event occurs
void    tk_ev_wake(void * event);                      // wake tasks waiting for event
```

Creating a task

```
TK_OBJECT(to_keyboard); ----------------------------- in tk_ntask.h --------------- task *to_keyboard
TK_ENTRY(tk_keyboard); ------------------------------ in tk_ntask.h --------------- int   tk_keyboard( int parm )


e = TK_NEWTASK(&keyboardtask); ----------------- in osporttk.c ---------------- function adds keyboardtask descriptor
struct inet_taskinfo keyboardtask =                                              to tcb list.
                    {
                                        &to_keyboard,
                    "console",
                    tk_keyboard,
                    NET_PRIORITY   -  1,
                    IO_STACK_SIZE,
                    };

TK_ENTRY(tk_keyboard) ------------------------------ in tk_ntask.h --------------- int tk_keayboard( int parm )
{
   for (;;)
   {
      TK_SLEEP(1);   /* make keyboard yield some time */
      kbdio(); /* let Iniche menu routines poll for char */
      keyboard_wakes++; /* count wakeups */

      if (net_system_exit)
         break;
   }
   TK_RETURN_OK();
}
```

Task Stack Protection

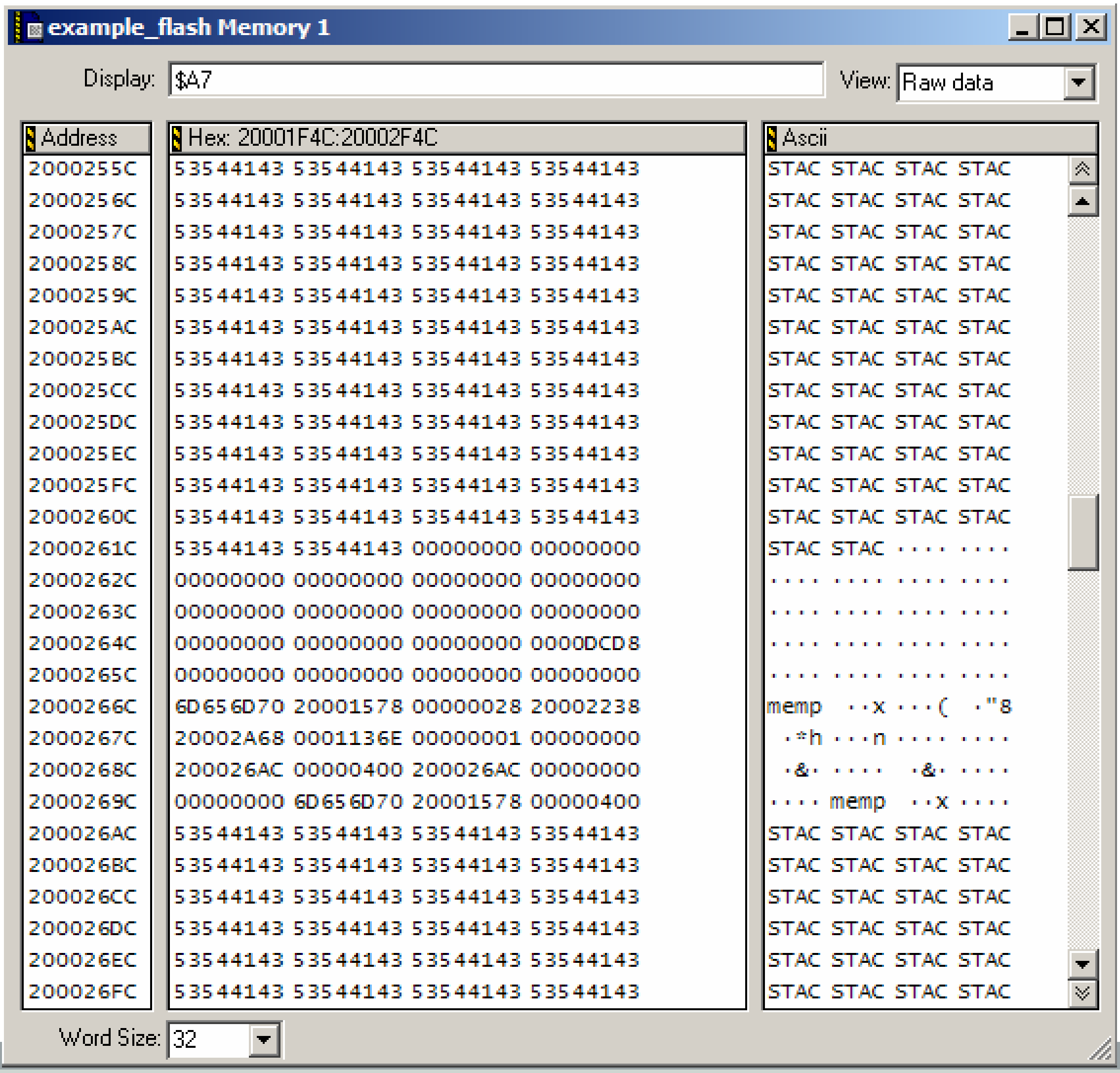
When a task is created, the stack space for that task in filled with a pattern ‘STAC’.

This pattern is used to indicate a fault if a stack overrun occurs. This is referred to as a guardband.

The tkstat command uses this pattern to determine the amount of stack used.

The guardband is checked at every task switch.

Task Stack Protection



example_flash Memory 1

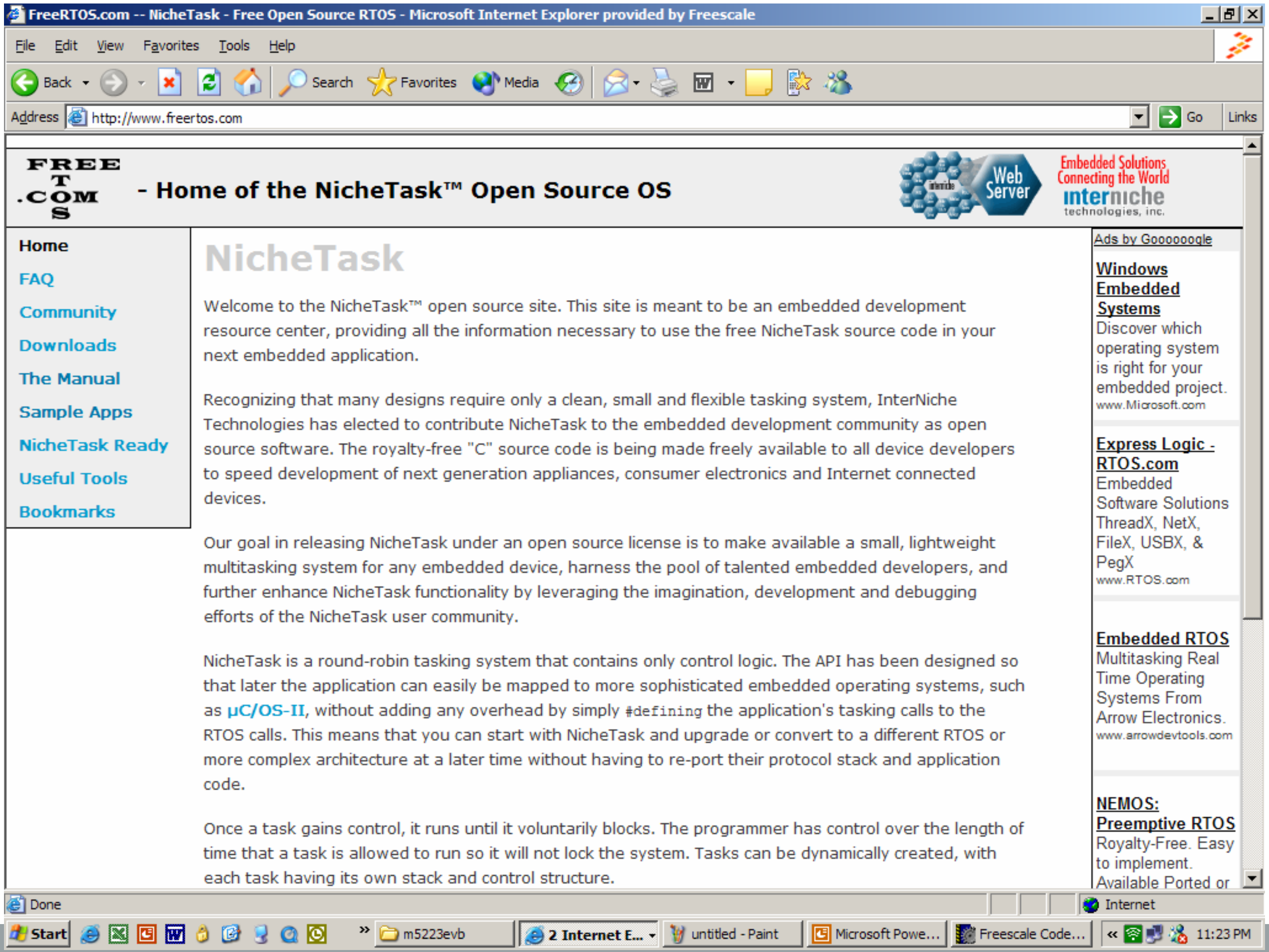
Display: $A7   View: Raw data

```
Address   Hex: 20001F4C:20002F4C                   Ascii
2000255C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000256C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000257C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000258C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000259C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025AC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025BC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025CC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025DC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025EC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200025FC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000260C  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
2000261C  53544143 53544143 00000000 00000000   STAC STAC ···· ····
2000262C  00000000 00000000 00000000 00000000   ···· ···· ···· ····
2000263C  00000000 00000000 00000000 00000000   ···· ···· ···· ····
2000264C  00000000 00000000 00000000 0000DCD8   ···· ···· ···· ····
2000265C  00000000 00000000 00000000 00000000   ···· ···· ···· ····
2000266C  6D656D70 20001578 00000028 20002238   memp  ··x ···(  ·"8
2000267C  20002A68 0001136E 00000001 00000000    ·*h ···n ···· ····
2000268C  200026AC 00000400 200026AC 00000000    ·&· ····  ·&· ····
2000269C  00000000 6D656D70 20001578 00000400   ···· memp  ··x ····
200026AC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200026BC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200026CC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200026DC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200026EC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
200026FC  53544143 53544143 53544143 53544143   STAC STAC STAC STAC
```

Word Size: 32

For More Information on the RTOS

http://www.freertos.com


FreeRTOS.com -- NicheTask - Free Open Source RTOS - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.freertos.com
Go Links
FREE RTOS .COM
Web Server
Embedded Solutions Connecting the World
interniche technologies, inc.
Home
FAQ
Community
Downloads
The Manual
Sample Apps
NicheTask Ready
Useful Tools
Bookmarks
Done
Internet
Start
m5223evb
2 Internet E...
untitled - Paint
Microsoft Powe...
Freescale Code...
11:23 PM

- Home of the NicheTask™ Open Source OS

NicheTask

Welcome to the NicheTask™ open source site. This site is meant to be an embedded development resource center, providing all the information necessary to use the free NicheTask source code in your next embedded application.

Recognizing that many designs require only a clean, small and flexible tasking system, InterNiche Technologies has elected to contribute NicheTask to the embedded development community as open source software. The royalty-free "C" source code is being made freely available to all device developers to speed development of next generation appliances, consumer electronics and Internet connected devices.

Our goal in releasing NicheTask under an open source license is to make available a small, lightweight multitasking system for any embedded device, harness the pool of talented embedded developers, and further enhance NicheTask functionality by leveraging the imagination, development and debugging efforts of the NicheTask user community.

NicheTask is a round-robin tasking system that contains only control logic. The API has been designed so that later the application can easily be mapped to more sophisticated embedded operating systems, such as μC/OS-II, without adding any overhead by simply `#defining` the application's tasking calls to the RTOS calls. This means that you can start with NicheTask and upgrade or convert to a different RTOS or more complex architecture at a later time without having to re-port their protocol stack and application code.

Once a task gains control, it runs until it voluntarily blocks. The programmer has control over the length of time that a task is allowed to run so it will not lock the system. Tasks can be dynamically created, with each task having its own stack and control structure.

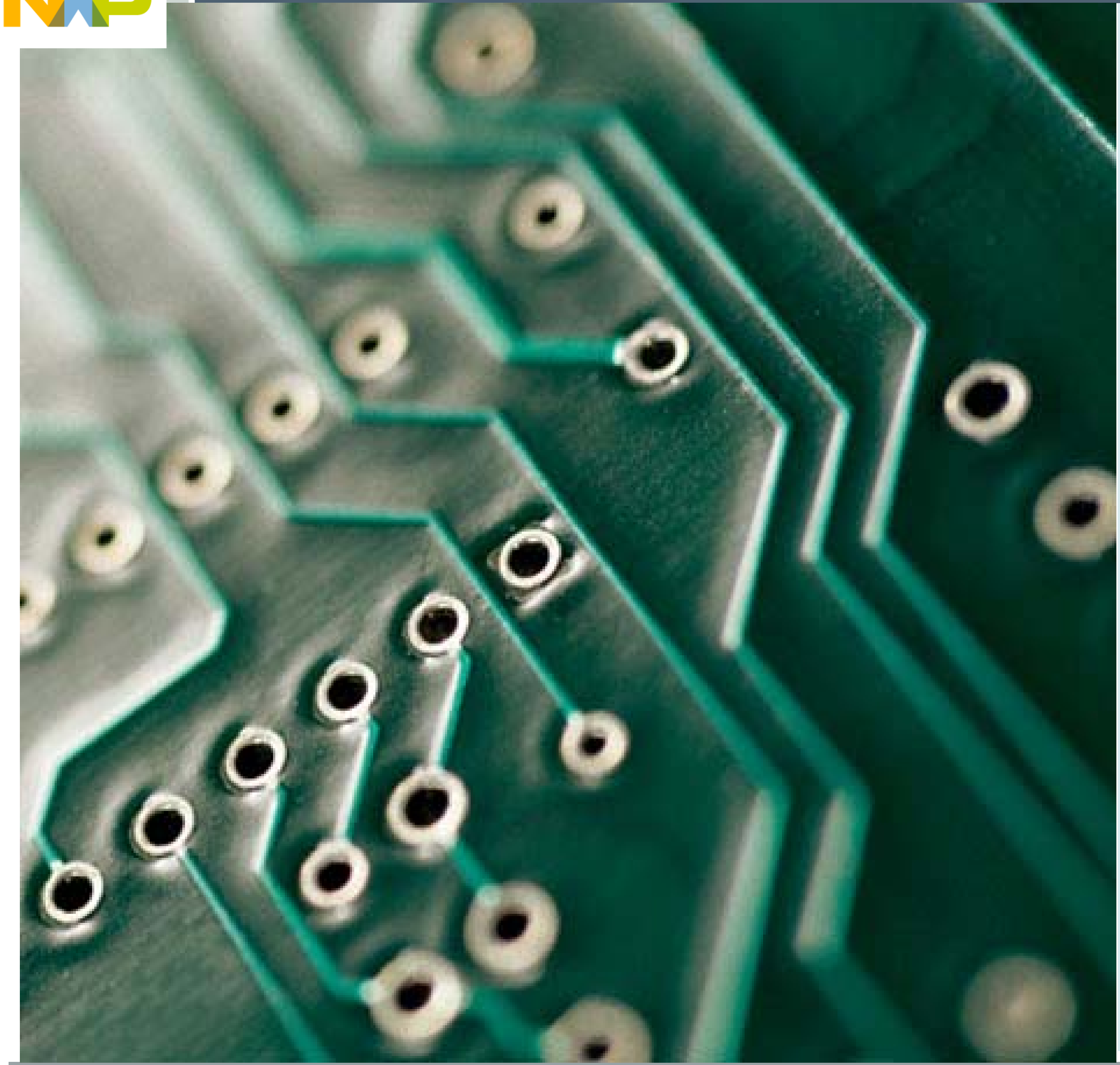
Ads by Goooooogle

Windows Embedded Systems
Discover which operating system is right for your embedded project.
www.Microsoft.com

Express Logic - RTOS.com
Embedded Software Solutions ThreadX, NetX, FileX, USBX, & PegX
www.RTOS.com

Embedded RTOS
Multitasking Real Time Operating Systems From Arrow Electronics.
www.arrowdevtools.com

NEMOS: Preemptive RTOS
Royalty-Free. Easy to implement. Available Ported or



LAB: RTOS

RTOS LAB

In this LAB we will load a simple project consisting of multiple tasks blinking LED’s.

We will experiment with altering the sleep times of each task and observing the results.

We will also show the result of a task not giving up real-time.

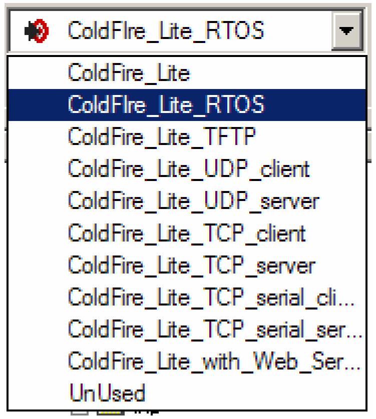
LAB

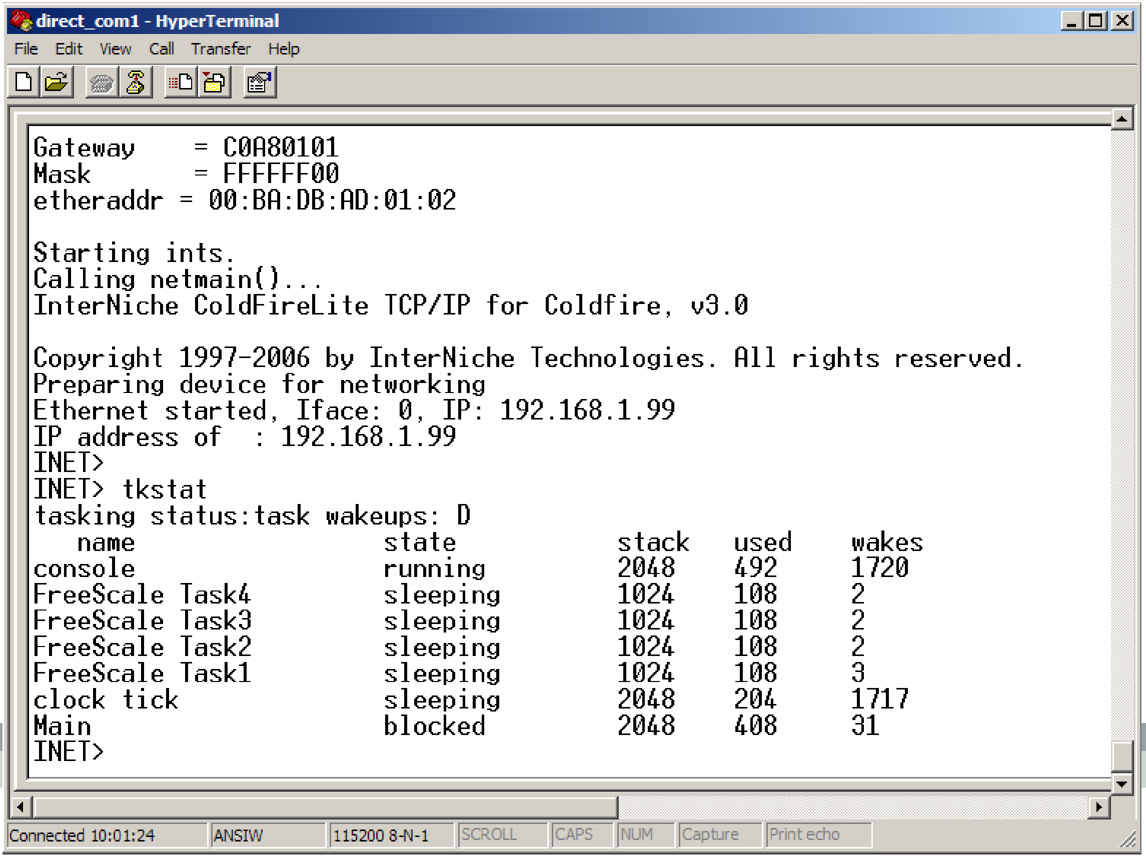
Flash the ColdFire_Lite_RTOS target

Run

Observe the LED’s

Type tkstat at the inet prompt.





```
Gateway    = C0A80101
Mask       = FFFFFF00
etheraddr = 00:BA:DB:AD:01:02

Starting ints.
Calling netmain()...
InterNiche ColdFireLite TCP/IP for Coldfire, v3.0

Copyright 1997-2006 by InterNiche Technologies. All rights reserved.
Preparing device for networking
Ethernet started, Iface: 0, IP: 192.168.1.99
IP address of  : 192.168.1.99
INET>
INET> tkstat
tasking status:task wakeups: D
   name                    state            stack   used    wakes
console                  running          2048    492     1720
FreeScale Task4          sleeping         1024    108     2
FreeScale Task3          sleeping         1024    108     2
FreeScale Task2          sleeping         1024    108     2
FreeScale Task1          sleeping         1024    108     3
clock tick               sleeping         2048    204     1717
Main                     blocked          2048    408     31
INET>
```

What’s happening

For this lab, 4 tasks are created.
Each task blinks a LED at a different rate.
The rate is controlled by the TK_SLEEP() in 5ms units.

Code in: freescale_RTOS.c

```
TK_ENTRY(tk_freescale4)
{
                int             i;

                // Wait for TCP/IP stack to init
                while (!iniche_net_ready)
                TK_SLEEP(1);

                // Task's must not return, Infinite loops
                for (;;)
                {
#if 1
                                // Good
                                TK_SLEEP(1400);
//                              printf( "\nTask4" );
#else
                                // Bad
                                for( i=0; i<0xFFFF; i++ );
#endif

                                LED3_TOGGLE;

                if(net_system_exit)
                break;
                }
                TK_RETURN_OK();
}
```

What happens if a task does NOT sleep

1) Comment out a TK_SLEEP() in one of the tasks.

Rebuild, flash, and run.

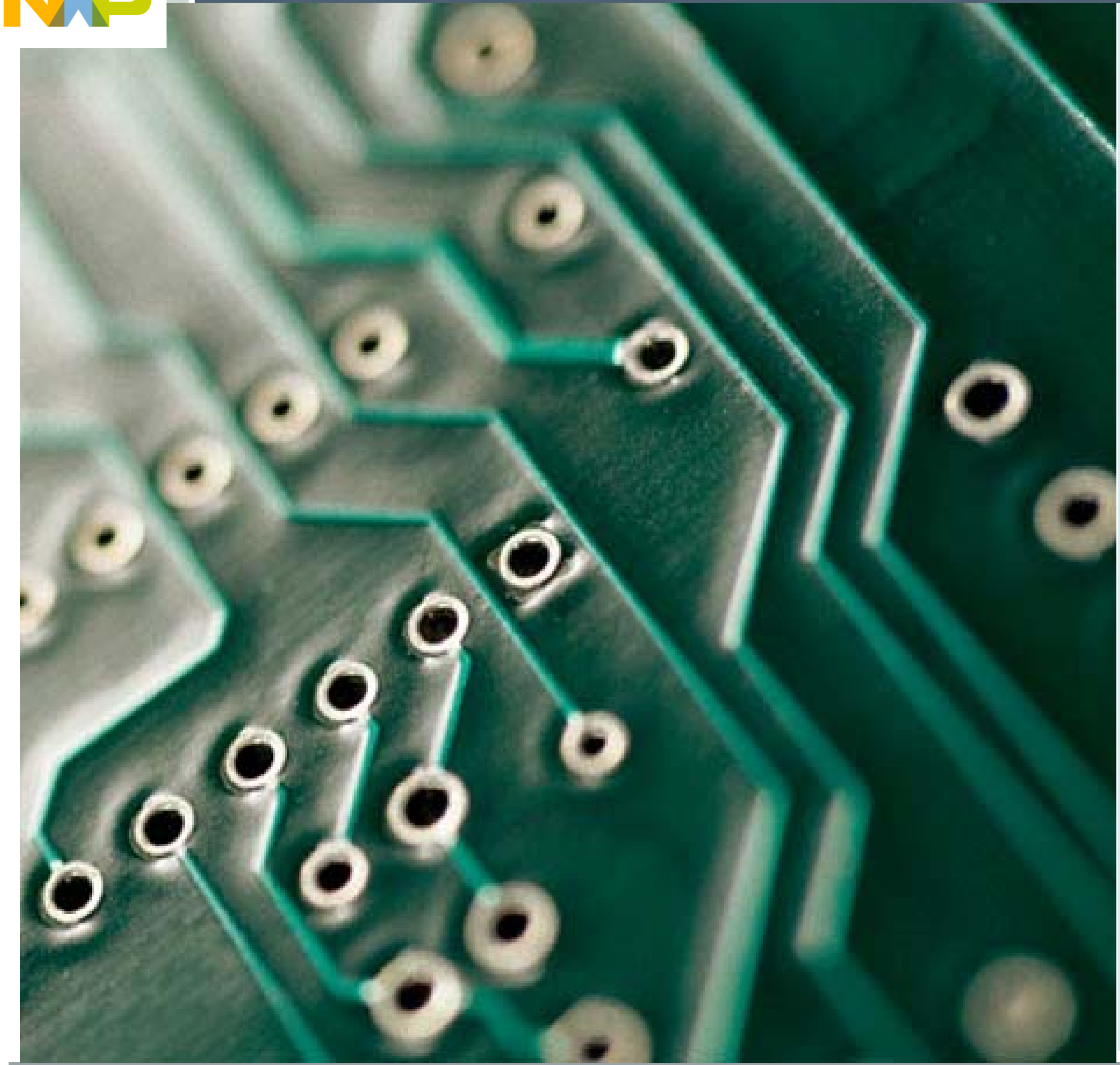
Are all 4 tasks still running?

Can you type anything at the console prompt?

2) Add a printf() to the task (still no sleep).

Rebuild, flash, and run.

What happens?



CodeWarrior 6.3 features

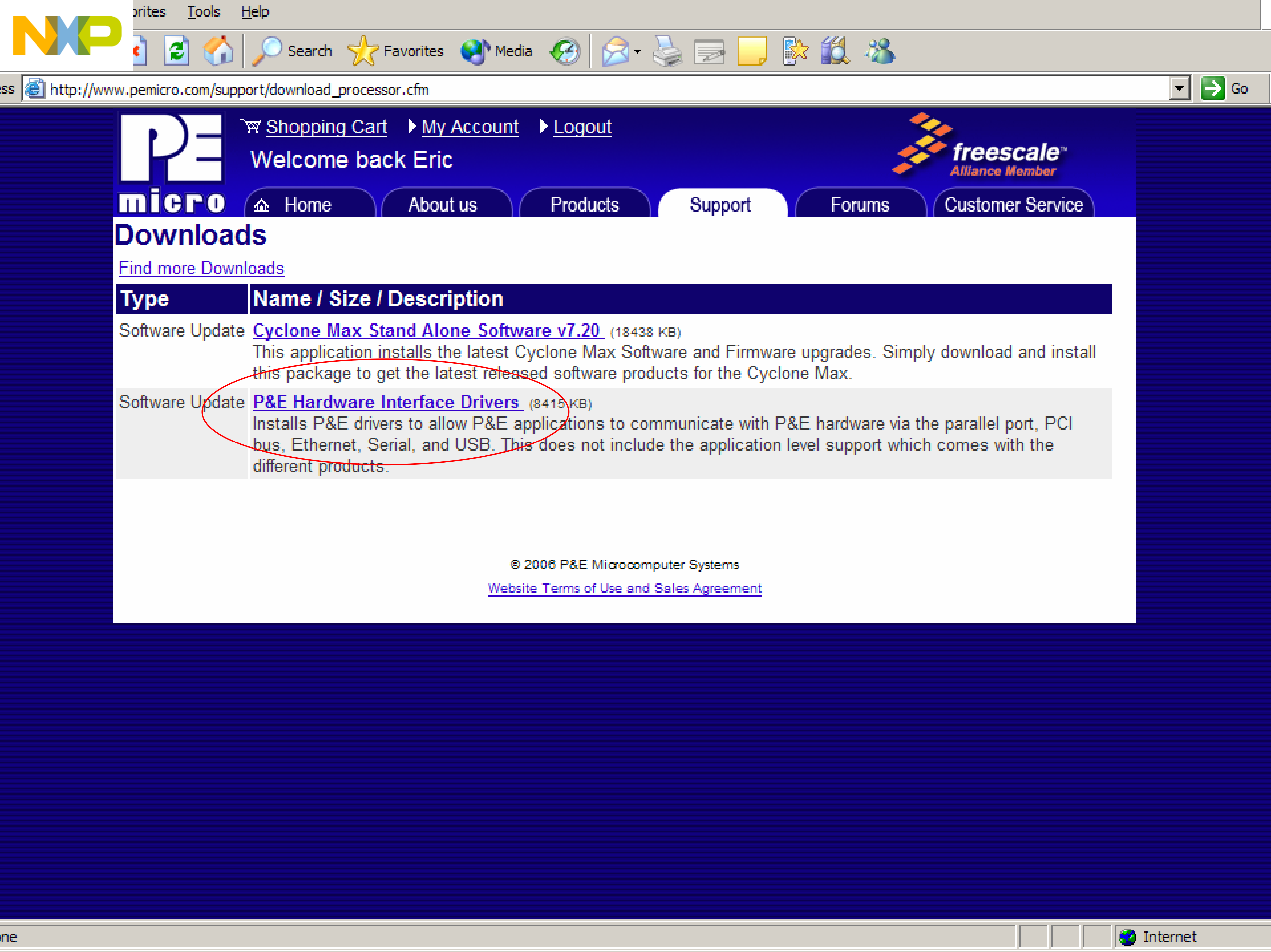
Issue when upgrading from 6.3 preview edition

Many customer had problems upgrading from the preview edition to the special edition of CodeWarrior 6.3.

This problem appeared primarily when customers also had CodeWarrior for S08 installed.

The issue is with the P&E drivers for the USB debugger.

Installing the new P&E drivers from the P&E site was the simplest method of resolving this issue.



Welcome back Eric

Downloads

Find more Downloads

Type	Name / Size / Description
Software Update	Cyclone Max Stand Alone Software v7.20 (18438 KB) This application installs the latest Cyclone Max Software and Firmware upgrades. Simply download and install this package to get the latest released software products for the Cyclone Max.
Software Update	P&E Hardware Interface Drivers (8415 KB) Installs P&E drivers to allow P&E applications to communicate with P&E hardware via the parallel port, PCI bus, Ethernet, Serial, and USB. This does not include the application level support which comes with the different products.

© 2006 P&E Microcomputer Systems

Website Terms of Use and Sales Agreement

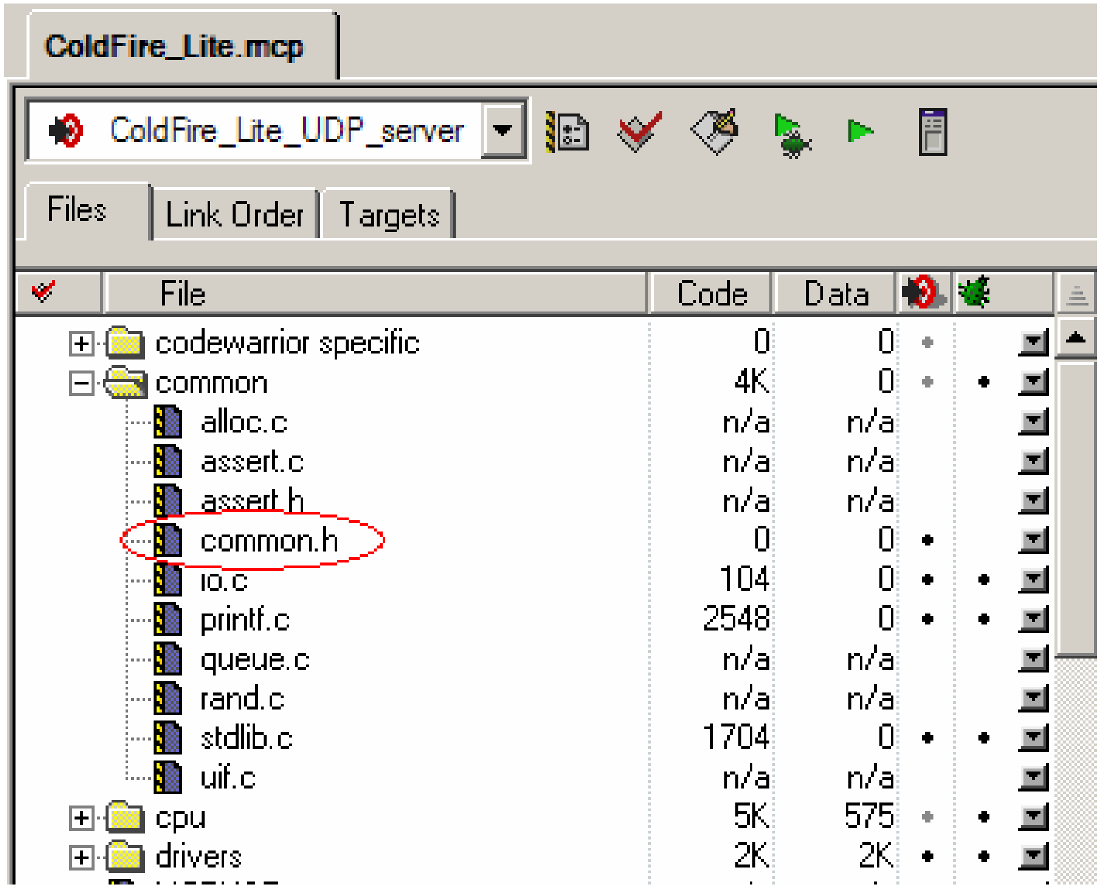
Issue building TCP/IP stack with 6.3 preview edition

Something changed with the release of CodeWarrior 6.3 causing many errors when building.

This issue was traced to common.h.

It has been fixed in this rev of the stack. For customers using the older version of the stack, make the following simple change.

Open common.h


ColdFire_Lite.mcp

ColdFire_Lite_UDP_server

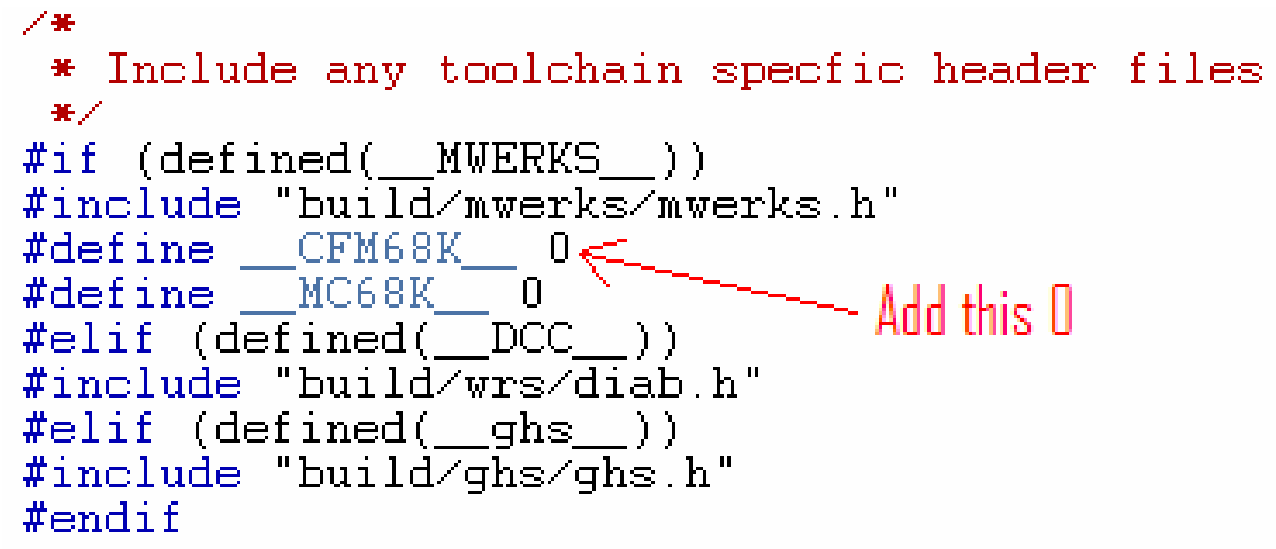
Files | Link Order | Targets

✓	File	Code	Data		
	codewarrior specific	0	0	•	
	common	4K	0	•	•
	alloc.c	n/a	n/a		
	assert.c	n/a	n/a		
	assert.h	n/a	n/a		
	common.h	0	0	•	
	io.c	104	0	•	•
	printf.c	2548	0	•	•
	queue.c	n/a	n/a		
	rand.c	n/a	n/a		
	stdlib.c	1704	0	•	•
	uif.c	n/a	n/a		
	cpu	5K	575	•	•
	drivers	2K	2K	•	•

Modify common.h

Set __CFM68K__ to 0

And __MC68K__ to 0



```
/*
 * Include any toolchain specfic header files
 */
#if (defined(__MWERKS__))
#include "build/mwerks/mwerks.h"
#define __CFM68K__  0
#define __MC68K__  0
#elif (defined(__DCC__))
#include "build/wrs/diab.h"
#elif (defined(__ghs__))
#include "build/ghs/ghs.h"
#endif
```

CodeWarrior 6.3 handles Interrupts differently

The ColdFire supports up to 8 interrupt levels, 0 to 7.

Normally this can be used to support prioritized nested interrupts.

CodeWarrior 6.3 disable this by default.

The normal method of writing a ISR

```
;  816: __declspec(interrupt)
;  817: void
;  818: fec_isr(void)
;  819: {
;  820:    BD *     bdp;
;  821:    int      i;
;  822:
;
0x00000000                  _fec_isr:
;                           fec_isr:
0x00000000  0x40E746FC2700         strldsr  #0x2700 <- Disables interrupts
0x00000006  0x4E560000           link     a6,#0
0x0000000A  0x4FEFFFC4           lea      -60(a7),a7
0x0000000E  0x48EF33FF000C         movem.l  d0-d7/a0-a1/a4-a5,12(a7)
;
```

Overriding this “feature”

```
;  816: __declspec(interrupt:0)
;  817: void
;  818: fec_isr(void)
;  819: {
;  820:    BD *     bdp;
;  821:    int      i;
;  822:
;
0x00000000                   _fec_isr:
;                       fec_isr:                  Note: interrupts NOT disabled
0x00000000  0x4E560000             link     a6,#0
0x00000004  0x4FEFFFC4              lea      -60(a7),a7
0x00000008  0x48EF33FF000C           movem.l  d0-d7/a0-a1/a4-
a5,12(a7)
```

Forcing the interrupt level to another value

```
;  816: __declspec(interrupt:0x2200)
;  817: void
;  818: fec_isr(void)
;  819: {
;  820:     BD *      bdp;
;  821:     int       i;
;  822:
;
0x00000000                          _fec_isr:
;                              fec_isr:
0x00000000  0x40E746FC2200          strldsr  #0x2200 <- SR = 0x2200
```

Why bother

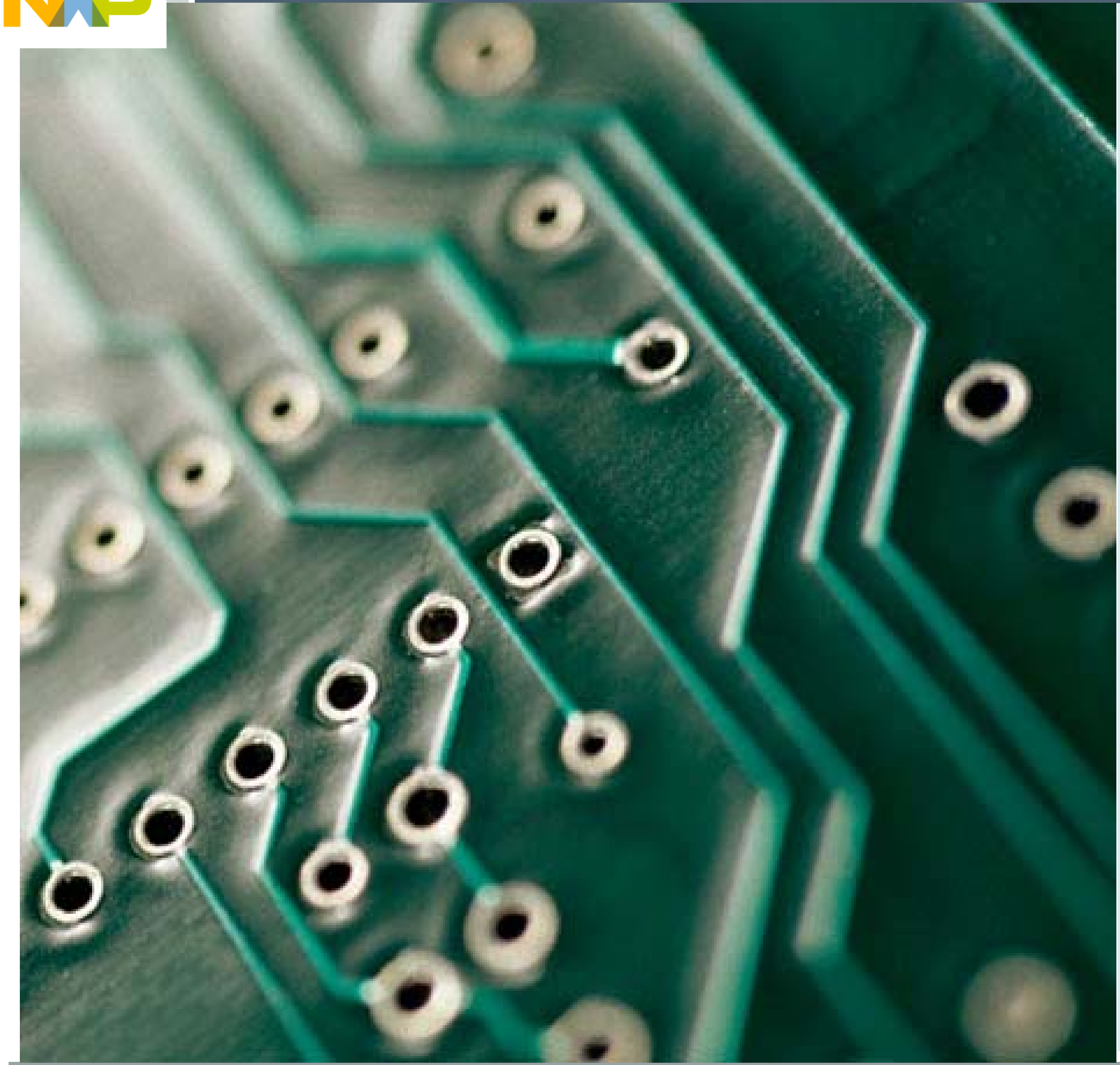
Many customers require a low latency on servicing a specific interrupt.

Traditionally you would assign your “must do now” interrupt to the highest maskable level (6).

By disabling interrupts upon entering ANY isr (even a lower priority ISR) you get what’s refered to as a priority inversion. The lower priority interrupt holds off the higher priority interrupt.

Example: 802.15.4 MAC port

The 802.15.4 MAC interrupt MUST happen within 8us of the external IRQ signal.
The FEC ISR can run for over 100us.
The FEC ISR is set to a lower level then the MAC IRQ.



The Serial Port Driver

The serial driver

The serial driver is initialized in the function mcf5223_init() in the file mcf5223_sysinit.c.
Uart_init() is in the file iuart.c
The driver supports all three of the serial ports.

The define POLLED_UART controls the mode of the driver.
#ifdef POLLED_UART

- Puts the UART driver in polled mode.

#ifndef POLLED_UART

- Puts the UART driver in interrupt mode.

The serial driver parameters

In the file iuart.c

Here the serial RX buf is limited to 32 bytes, the TX buf is 256 bytes.

```
#ifndef UART_RXBUFSIZE
#define UART_RXBUFSIZE        32 // EMG
#endif

#ifndef UART_TXBUFSIZE
#define UART_TXBUFSIZE        256              // EMG - I decreased this
#endif

#ifndef UART0_SPEED
#define UART0_SPEED           115200
#endif

#ifndef UART1_SPEED
#define UART1_SPEED           19200
#endif
```

The Serial Console Interface – type help at the INET> prompt

```
INET> help
SNMP Station: general commands:
   help     - help with menus
   state    - show current station setup
   delay    - set milliseconds to wait between pings
   host     - set default active IP host
   length   - set default ping packet length
   quit     - quit station program
   ping     - send a ping
   baud     - set serial console BAUD
   setip    - set interface IP address
   version  - display version information
   !command - pass command to OS shell
Also try 'help [general|diagnostic|EMG HTTP]'
INET>
```

Type help diag at the INET> prompt

```
INET> help diag
SNMP Station: diagnostic commands:
   arps      - display ARP stats and table
   buffers  - display free q buffer stats
   queues   - dump packet buffer queues
   dbytes   - dump block of memory
   debug    - set IP stack debug tracing
   dtrap    - try to hook debugger
   iface    - display net interface stats
   linkstats - display link layer specific stats
   tcp      - display TCP stats
   sockets  - display socket list
   tbconn   - tcp BSD connection stats
   tbsend   - tcp BSD send stats
   tbrcv    - tcp BSD receive stats
   allocsize - set size for alloc() breakpoint
   ipstat   - display IP layer stats
   icmpstat - display ICMP layer stats
   udp      - display UDP layer stats
   upcall   - trace received packets
   tkstats  - tasking system status
   users    - list all users
   adduser  - add a new user
INET>
```

HTTP Server Info

```
INET> help EMG
SNMP Station: EMG HTTP commands:
   dir      - Dir of EMG FFS
   flash_erase - Erase the dynamic FLASH area
   var      - Dynamic HTML variable
   http     - Dump HTTP sessions array
INET> http


HTTP sessions array Dump

STATE    VALID    KEEP_ALIVE    FILE_POINTER    SOCKET
Wait for header    Not Valid    0    0x0    0x0
Wait for header    Not Valid    0    0x0    0x0
Wait for header    Not Valid    0    0x0    0x0
Wait for header    Not Valid    0    0x0    0x0

INET>
```

Insight into the RTOS

```
INET> tkstats
tasking status:task wakeups: D
   name                    state        stack   used    wakes
console                    running       2048    536     1216676
EMG HTTP server   ready           2048    192     51859563
clock tick                 sleeping      2048    104     42047
Main                       blocked       4096    392     0
INET>
```

Ethernet info – the iface command

```
INET> iface
Interface  - Fast Ethernet
Status; Admin:up Oper:up for: 8 minutes, 45 sec.
rcvd: errors:0   dropped:0   station:0   bcast:0   bytes:0
sent: errors:0   dropped:0   station:0   bcast:0   bytes:0
MAC address: 00 CF 52 23 00 00 ..R#..



Control Register                = 3000

DATARATE = 100Mbps
ANE = Autonegotiation Enabled
DPLX = Half Duplexe

This register advertises the capabilities of the port to the MII
Status Register                 = 7849

Indicates the PHY supports 100BASE-TX full-duplex mode
Indicates the PHY supports 100BASE-TX half-duplex mode
Indicates the PHY supports 10BASE-T full-duplex mode
Indicates the PHY supports 10BASE-T half-duplex mode
No fault detected
PHY has auto-negotiation ability
valid link has NOT been established
AutoNegotiation NOT complete - Data is NOT Valid

Auto-Neg. Advertisement Register = 81E1

100BASE-TX full -duplex capable
100BASE-TX half-duplex capable
10BASE-T full-duplex capable
10BASE-T half-duplex capable

INET>
```

Adding a Your Own Command

```
//******************************************************************************
// Fill out structure for EMG FFS DIRectory menu command
//******************************************************************************
struct menu_op emg_ffs_dir_menu[] =           {
            "EMG HTTP",       stooges,            "EMG HTTP menu",
            "dir",            emg_ffs_dir,        "Dir of EMG FFS",
            "flash_erase",    flash_erase,        "Erase the dynamic FLASH area",
            "var",            emg_http_var,       "Dynamic HTML variable",
            "http",           emg_http_sessions,  "Dump HTTP sessions array",
            NULL,
                              };
```

```
// Install Menu item 'DIR' for EMG FFS
if( install_menu( emg_ffs_dir_menu ) )
        printf( "\nCould not install DIR menu item for EMG FFS" );
```

Commands are passed arguments

```
//*********************************************************************************
// int SoftEthernetNegotiation( int seconds )   Written By Eric Gregori
//
// Work-around for bug in hardware autonegotiation.
// Attempt to connect at 100Mbps - Half Duplexe
// Wait for seconds
// Attempt to connect at 10Mbps - Half Duplexe
//
// Returns 10, or 100 on success, 0 on failure
//*********************************************************************************
int   set_baud(void * pio)
{
        char      *cp;


        cp = nextarg(((GEN_IO)pio)->inbuf);
        iuart_set_baud( 0, atoi(cp) );

        return(0);
}
```

Printf is supported with formatting

```
//*******************************************************************************
// Print Directory of Static and Dynamic Flash File Systems.
//
// Author: Eric Gregori  (847) 651 - 1971
//*******************************************************************************
int emg_ffs_dir(void * pio)
{
		int						file_count, total_file_size, k, j;
		volatile unsigned long			*fat_file_sys;
		volatile unsigned char			*fat_file_names;


		ns_printf( pio, "\nStatic FFS" );
		ns_printf( pio, "\n\n%-32s %-6s %-8s",
									"FILENAME",
									"LENGTH",
									" POINTER" );

		total_file_size = 0;

		// Loop through each file printing the info
		for( file_count=0; file_count<emg_static_ffs_nof; file_count++ )
		{
				ns_printf( pio, "\n%-33s", emg_static_ffs_filenames[file_count] );
				ns_printf( pio, "%-9d", emg_static_ffs_len[file_count] );
				ns_printf( pio, "0x%-8x", (unsigned long)emg_static_ffs_ptrs[file_count] );
				total_file_size += emg_static_ffs_len[file_count];
		}

		ns_printf(pio,"\n\n				Total Size = %d",total_file_size);
		ns_printf(pio,"\ntotal static files = %d\n",file_count);

		ns_printf( pio, "\nDynamic FFS" );
		ns_printf( pio, "\n\n%-32s %-6s %-8s",
									"FILENAME",
									"LENGTH",
									" POINTER" );
```

This package is ideal for remote testing

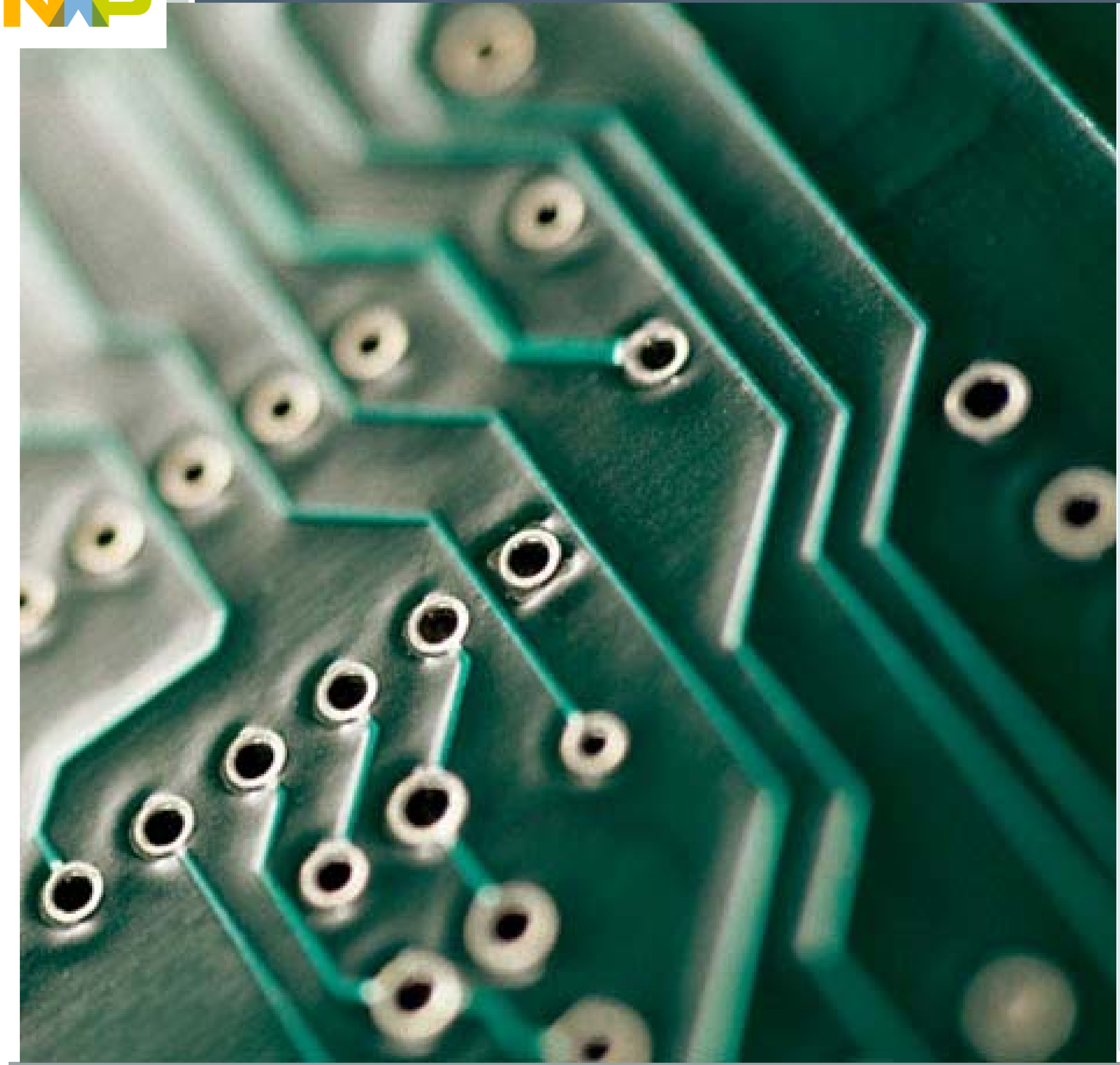
Imagine this, you need a method to instrument a device you are testing.

Just write your own command, or better yet put your data in a VAR, and you can access that data from anywhere in the world.

This is a ideal platform for engineers to write small test programs, or build quick prototypes.

The MCF5223 has:

- 2 independent 4 channel 12 bit A/D converters
- 8 PWM modules
- 4   24 bit timers (can be used as pulse accumulators)
- 1 16 bit timer
- IIC, SPI, 3 UARTS, …..



TFTP

TFTP

The Interniche stack includes a TFTP server and client.

TFTP requires a file system. Interniche also provides a RAM based virtual file system.

Since the 5223 only has 32K of RAM, the TFTP client and server can only be used for demo purposes.

The TFTP client/server have not yet been linked to the Flash File System.

TFTP project files

The TFTP project files are in the TFTP folder.

Tftpcli.c

- TFTP client

Tftpsrv.c

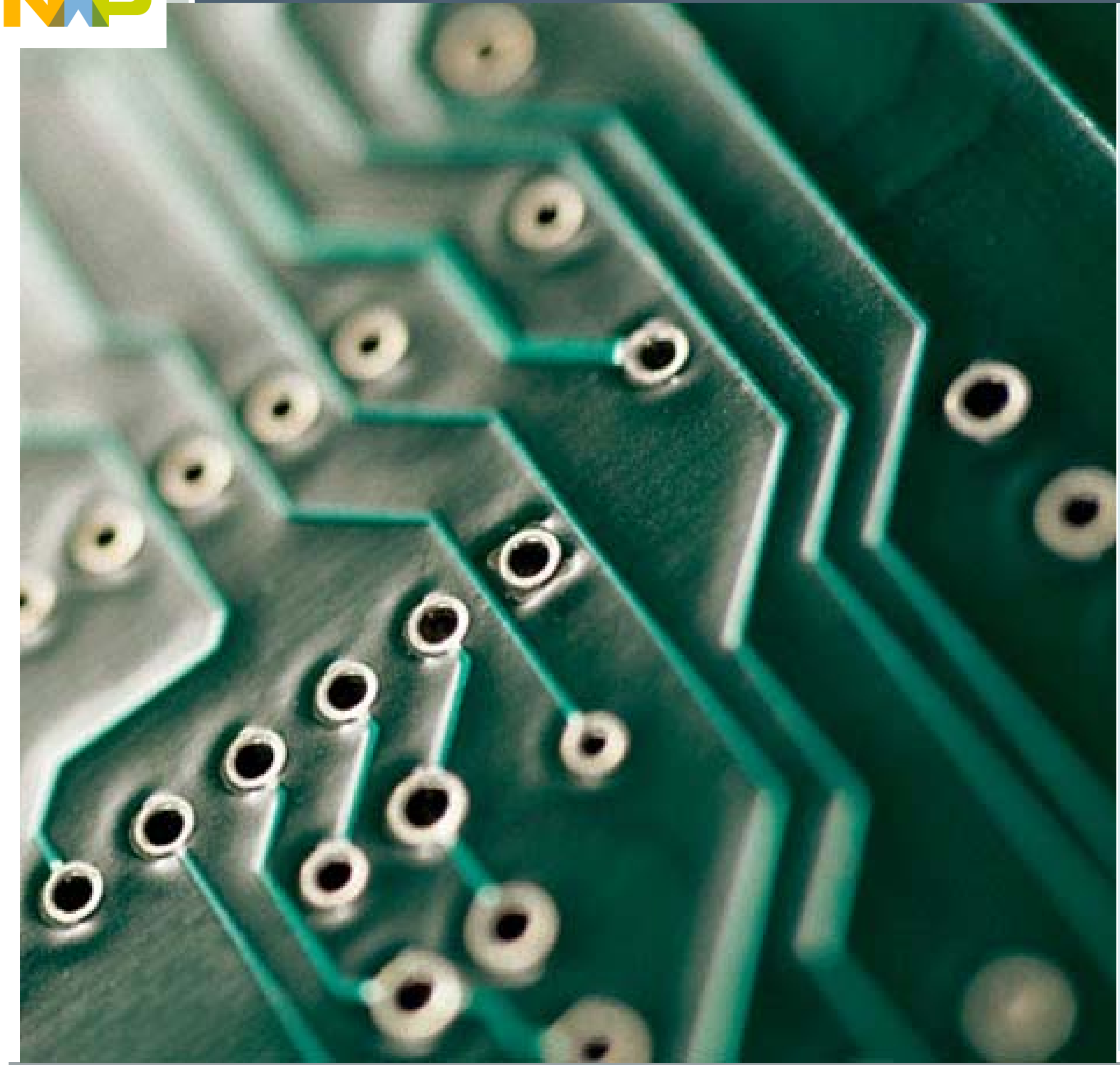
- TFTP server

Tftpudp.c

- This module contains the low-level UDP routines.

UDP

The module m_udp.c contains the UDP API.



LAB: TFTP

TFTP LAB

In this lab we will build a project with the TFTP client.

We will use the client to connect to the PC

First you must disable 2 services on your computer.

Blackd – BlackIce

And

DefWatch

These processes interfere with the UDP traffic that TFTP uses.

TFTP lab

Stop the DefWatch and Blackd processes on your machine.

Double click on the TFTPD32.exe

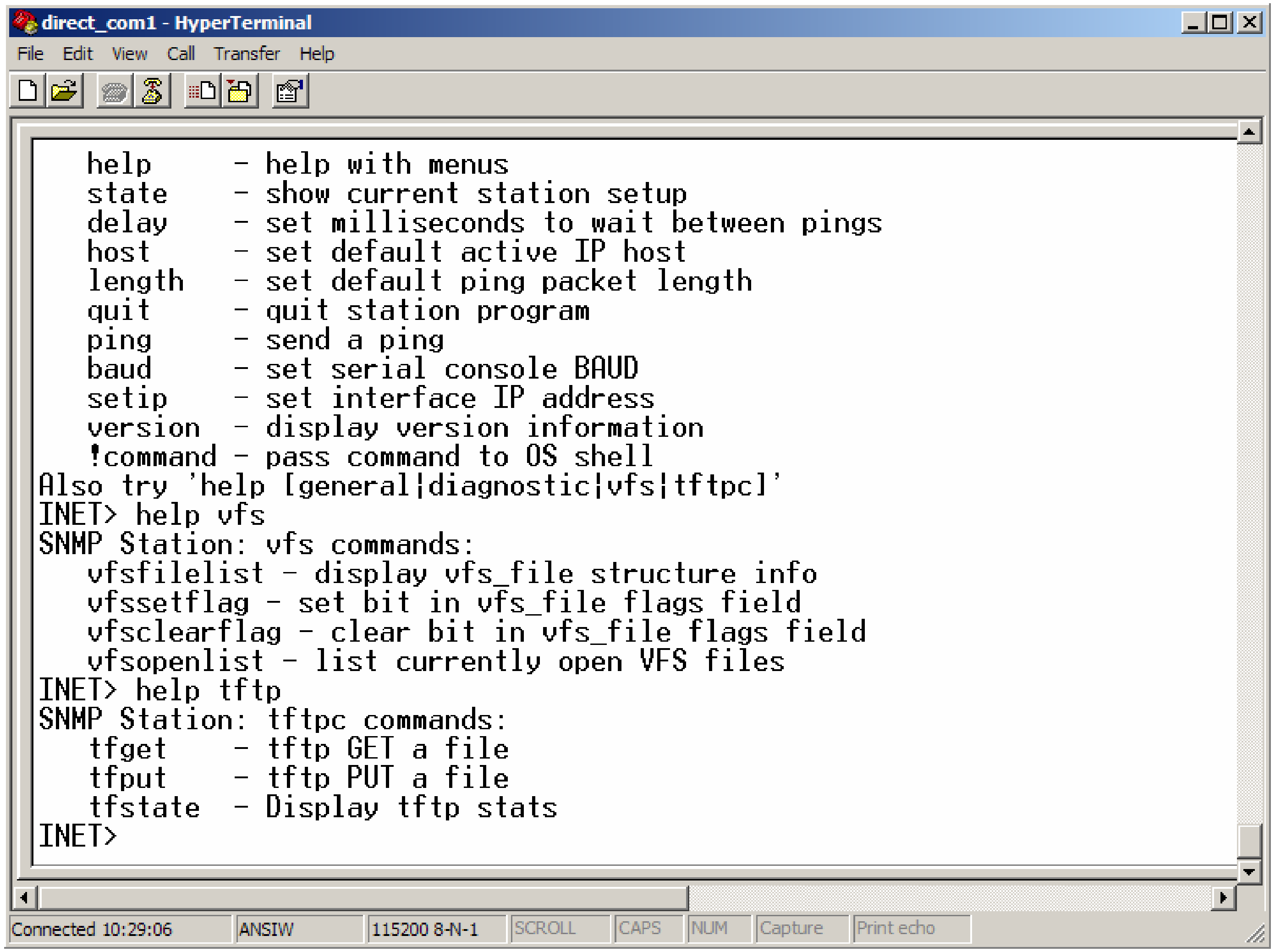
1) This opens the TFTP server on the PC.
2) I have included some small files.
    1) Test_file1.txt
    2) Test_file2.txt

At the INET> prompt type:

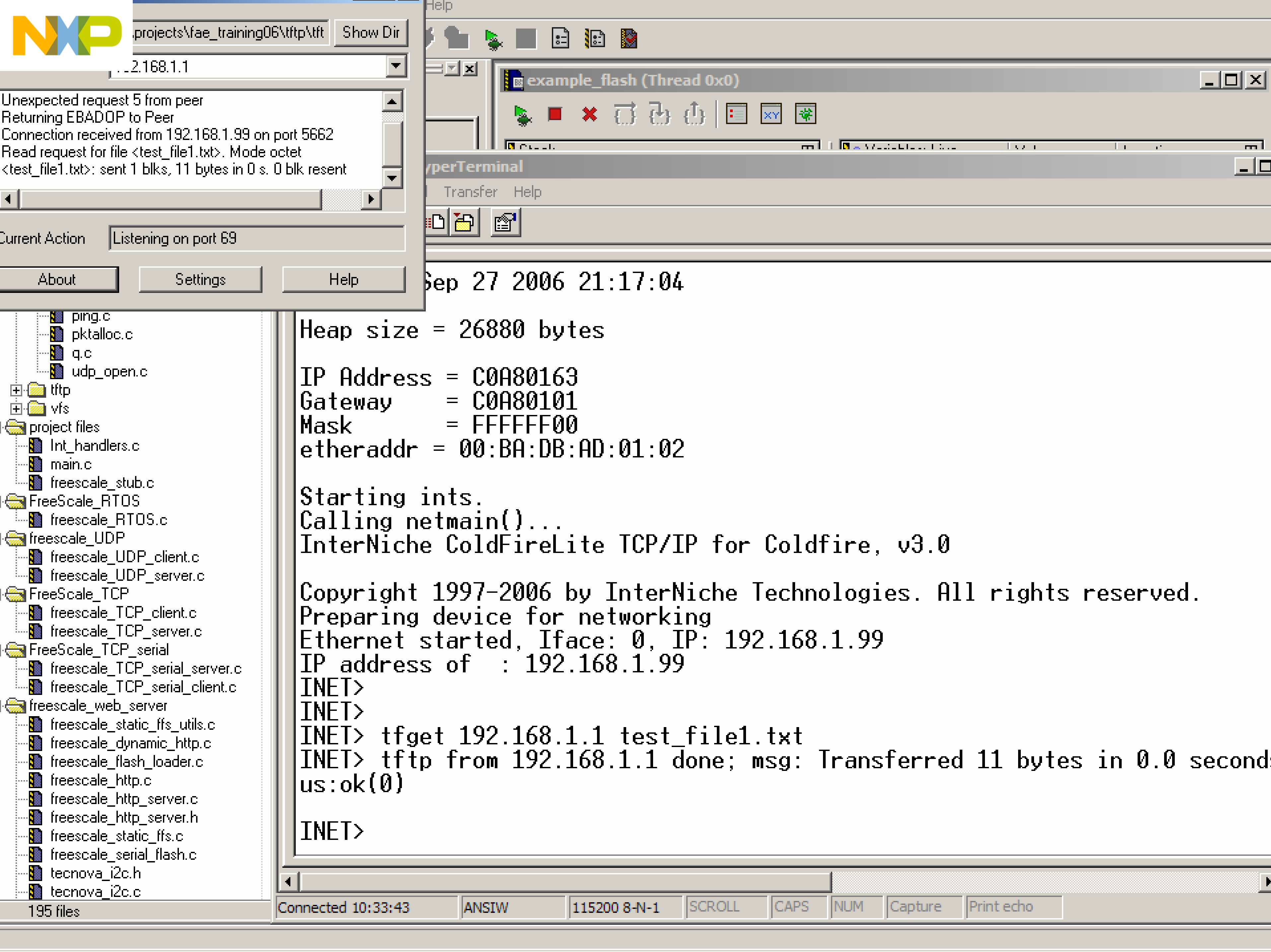
```
INET> vfsfilelist
total files = 0
dynamically allocated files = 0, buffer space = 0x0
INET>
```

Notice there are no files in the RAMdrive.

Additional commands with TFTP and VFS enabled



```
   help      - help with menus
   state     - show current station setup
   delay     - set milliseconds to wait between pings
   host      - set default active IP host
   length    - set default ping packet length
   quit      - quit station program
   ping      - send a ping
   baud      - set serial console BAUD
   setip     - set interface IP address
   version   - display version information
   !command  - pass command to OS shell
Also try 'help [general|diagnostic|vfs|tftpc]'
INET> help vfs
SNMP Station: vfs commands:
   vfsfilelist - display vfs_file structure info
   vfssetflag - set bit in vfs_file flags field
   vfsclearflag - clear bit in vfs_file flags field
   vfsopenlist - list currently open VFS files
INET> help tftp
SNMP Station: tftpc commands:
   tfget     - tftp GET a file
   tfput     - tftp PUT a file
   tfstate   - Display tftp stats
INET>
```



```
Unexpected request 5 from peer
Returning EBADOP to Peer
Connection received from 192.168.1.99 on port 5662
Read request for file <test_file1.txt>. Mode octet
<test_file1.txt>: sent 1 blks, 11 bytes in 0 s. 0 blk resent
```

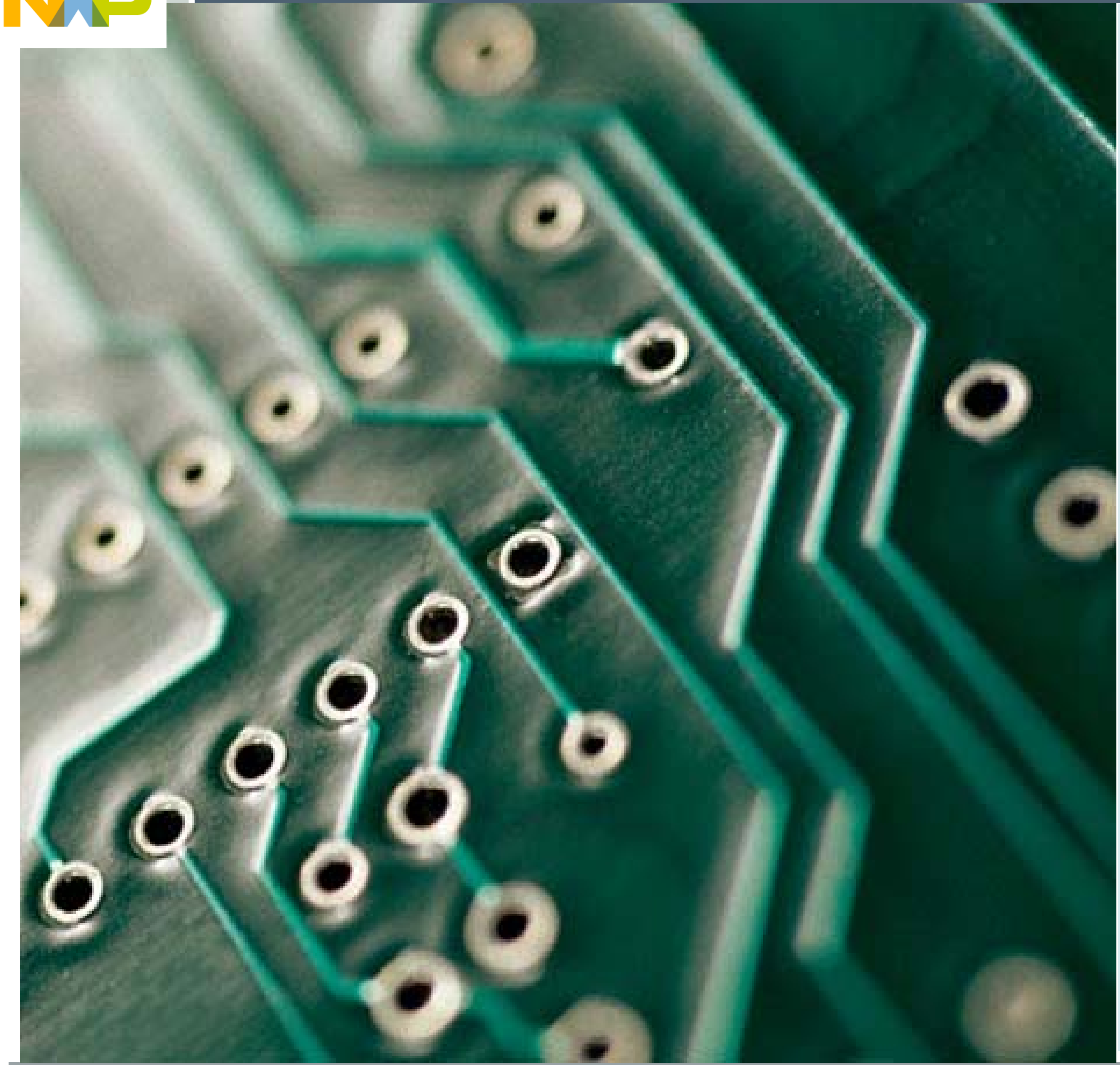
Current Action: Listening on port 69

```
ping.c
pktalloc.c
q.c
udp_open.c
tftp
vfs
project files
  Int_handlers.c
  main.c
  freescale_stub.c
FreeScale_RTOS
  freescale_RTOS.c
freescale_UDP
  freescale_UDP_client.c
  freescale_UDP_server.c
FreeScale_TCP
  freescale_TCP_client.c
  freescale_TCP_server.c
FreeScale_TCP_serial
  freescale_TCP_serial_server.c
  freescale_TCP_serial_client.c
freescale_web_server
  freescale_static_ffs_utils.c
  freescale_dynamic_http.c
  freescale_flash_loader.c
  freescale_http.c
  freescale_http_server.c
  freescale_http_server.h
  freescale_static_ffs.c
  freescale_serial_flash.c
  tecnova_i2c.h
  tecnova_i2c.c
```

```
Sep 27 2006 21:17:04

Heap size = 26880 bytes

IP Address = C0A80163
Gateway    = C0A80101
Mask       = FFFFFF00
etheraddr = 00:BA:DB:AD:01:02

Starting ints.
Calling netmain()...
InterNiche ColdFireLite TCP/IP for Coldfire, v3.0

Copyright 1997-2006 by InterNiche Technologies. All rights reserved.
Preparing device for networking
Ethernet started, Iface: 0, IP: 192.168.1.99
IP address of   : 192.168.1.99
INET>
INET>
INET> tfget 192.168.1.1 test_file1.txt
INET> tftp from 192.168.1.1 done; msg: Transferred 11 bytes in 0.0 second
us:ok(0)

INET>
```

After the file is uploaded

```
INET> vfsfilelist
test_file1.txt   -----WIDNS- 2000643C        B        B      100
total files = 1
dynamically allocated files = 1, buffer space = 0x100
INET>
```



DHCP

DHCP Firmware

When DHCP is enabled, the TCP/IP stack cannot complete its initialization until after the DHCP transaction is complete.

The function netmain_init() in the module allports.c calls the function dhc_setup() in dhcsetup.c.

dhc_setup() runs the DHCP protocol which will contact the DHCP server to aquire a IP address and other network related data.

Enabling the DHCP client

In the file ipport.h you will find the following.

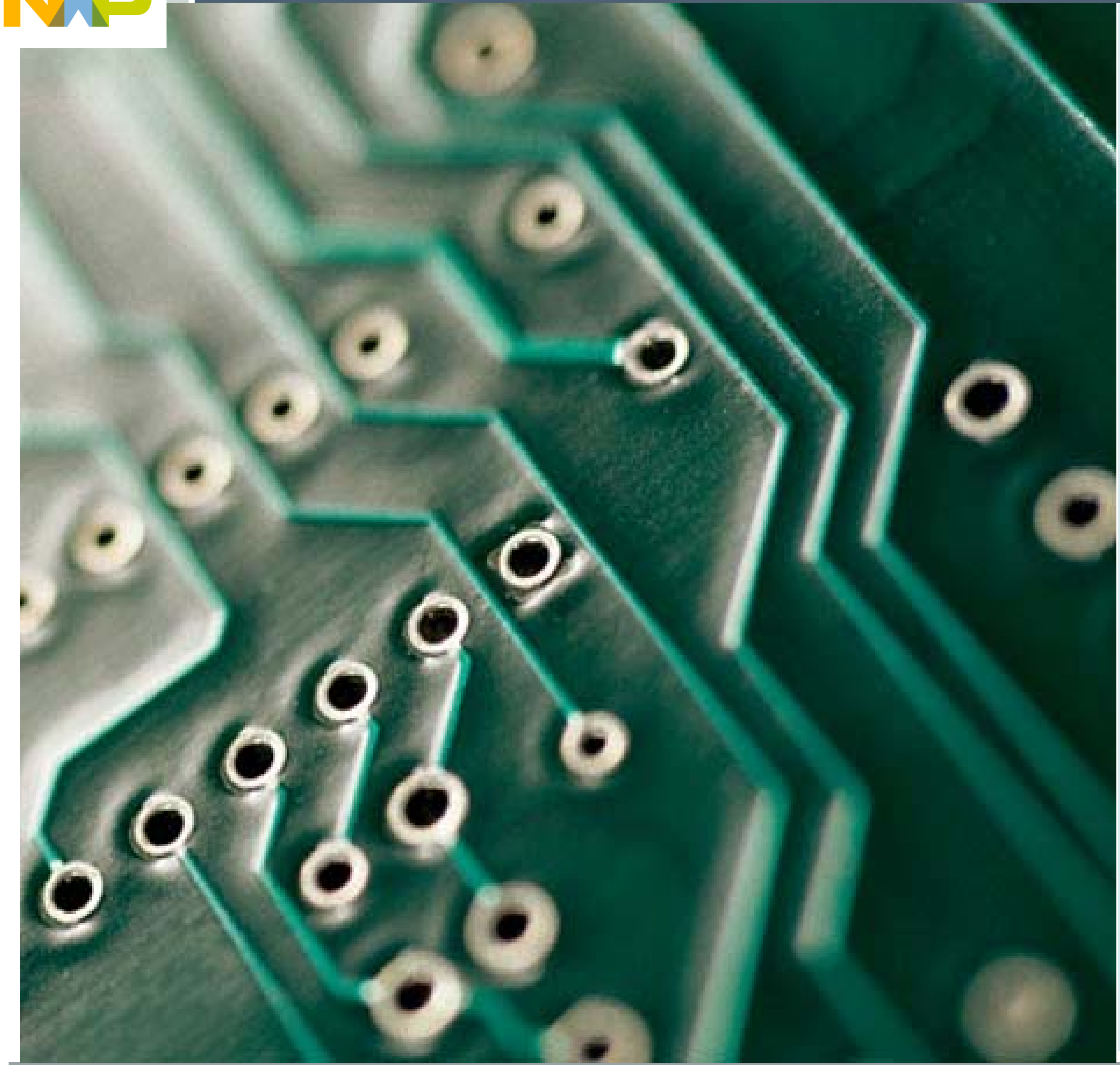
```
#define INCLUDE_ARP     1  /* use Ethernet ARP */
#define FULL_ICMP       1  /* use all ICMP || ping only */
#define OMIT_IPV4       1  /* not IPV4, use with MINI_IP */
#define MINI_IP         1   /* Use Nichelite mini-IP layer */
#define MINI_TCP        1   /* Use Nichelite mini-TCP layer */
#define MINI_PING       1   /* Build Light Weight Ping App for Niche Lite */
#define BSDISH_RECV     1   /* Include a BSD recv()-like routine with mini_tcp */
#define BSDISH_SEND     1   /* Include a BSD send()-like routine with mini_tcp */
#define NB_CONNECT      1  /* support Non-Blocking connects (TCP, PPP, et al) */
#define MUTE_WARNS      1  /* gen extra code to suppress compiler warnings */
#define IN_MENUS        1  /* support for InterNiche menu system */
#define NET_STATS       1  /* include statistics printfs */
#define QUEUE_CHECKING  1  /* include code to check critical queues */
#define INICHE_TASKS    1  /* InterNiche multitasking system */
#define MEM_BLOCKS      1  /* list memory heap stats */
// EMG #define TFTP_CLIENT     1  /* include TFTP client code */
// EMG #define TFTP_SERVER     1  /* include TFTP server code */
// EMG #define DNS_CLIENT      1  /* include DNS client code */
#define INICHE_TIMERS   1  /* Provide Interval timers */
```

// EMG - To enable DHCP, uncomment the line below
//#define DHCP_CLIENT   1 /* include DHCP client code */

```
// EMG #define INCLUDE_NVPARMS 1  /* non-volatile (NV) parameters logic */
#define NPDEBUG         1  /* turn on debugging dprintf()s */
// EMG #define VFS_FILES       1  /* include Virtual File System */
// EMG #define USE_MEMDEV      1   /* Psuedo VFS files mem and null */
#define NATIVE_PRINTF   1   /* use target build environment's printf function */
#define NATIVE_SPRINTF  1   /* use target build environment's printf function */
#define PRINTF_STDARG   1   /* build ...printf() using stdarg.h */
#define TK_STDIN_DEVICE 1   /* Include stdin (uart) console code */
#define BLOCKING_APPS   1   /* applications block rather than poll */
#define INCLUDE_TCP     1  /* this link will include NetPort TCP w/MIB */

/**** end of option list ***/
```

DHCP Demo

Pushing SW1 at power-up will enable DHCP.



UDP

UDP

UDP stands for User Datagram Protocol
It is a layer under the TCP layer in the TCP/IP stack.

UDP does not include acknowledgements or connections.

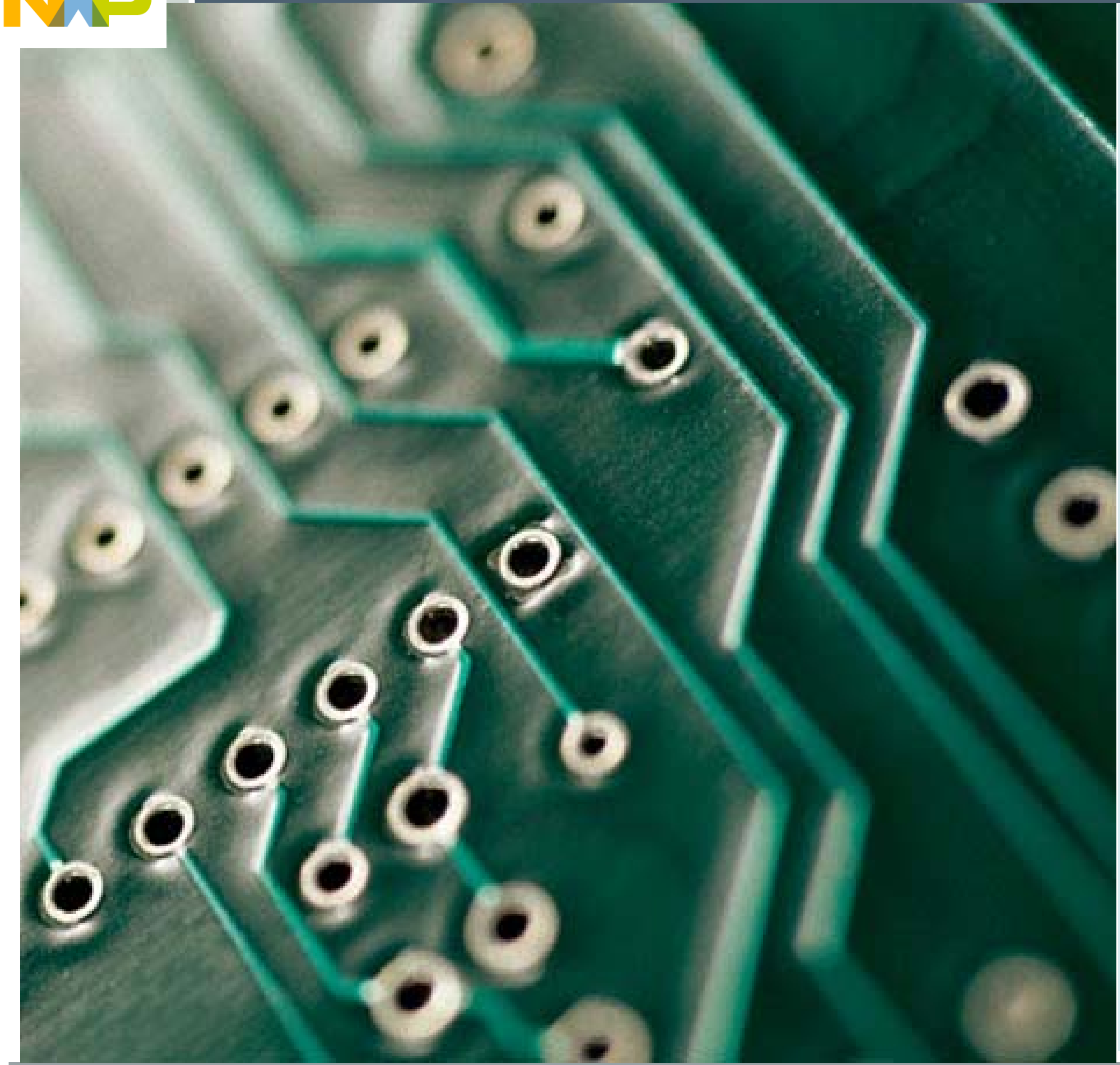
UDP does support 0xFFFD (65533) ports.

UDP is used whenever high speed data transfer is required.

UDP with ColdFire_Lite

Freescale_UDP_client.c contains an example of a UDP client.

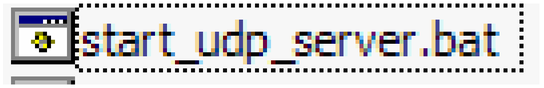
This client sends packets as fast as possible to the PC.

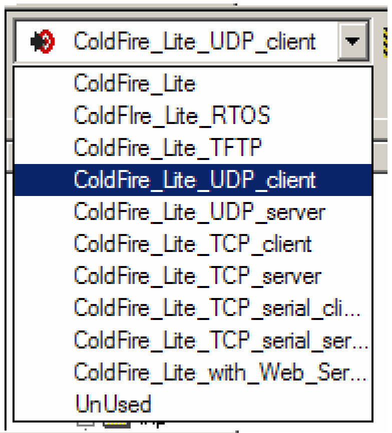


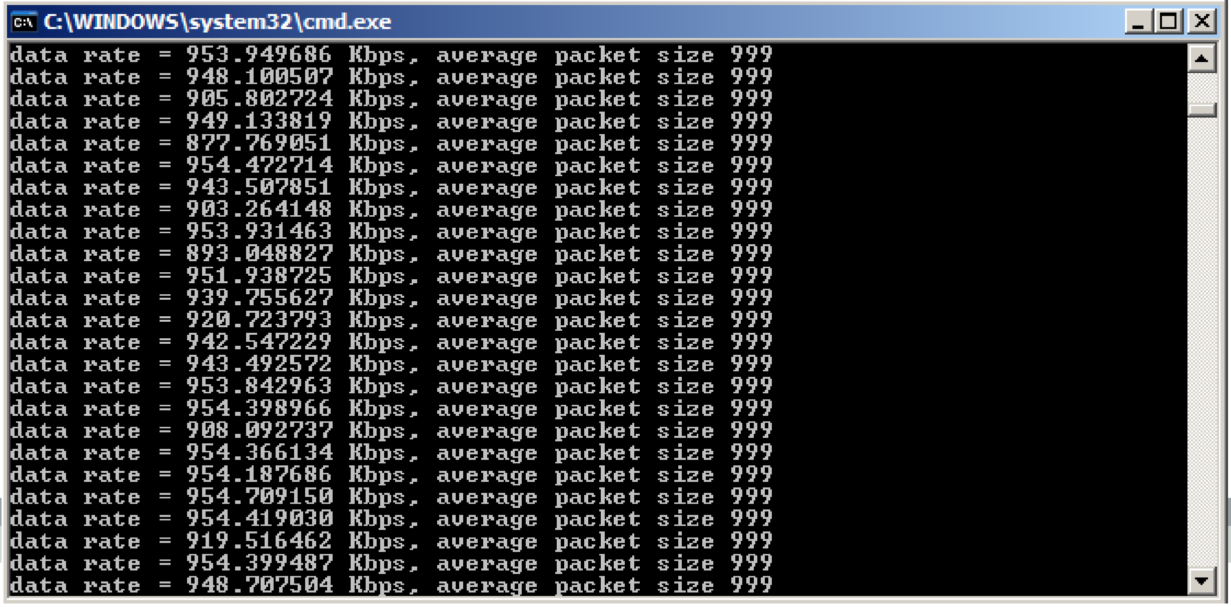
LAB: UDP client

LAB: UDP client

1) Select the ColdFire_Lite_UDP_client target
2) Build, flash, run
3) Execute the UDP server on the PC by clicking on the BAT file.





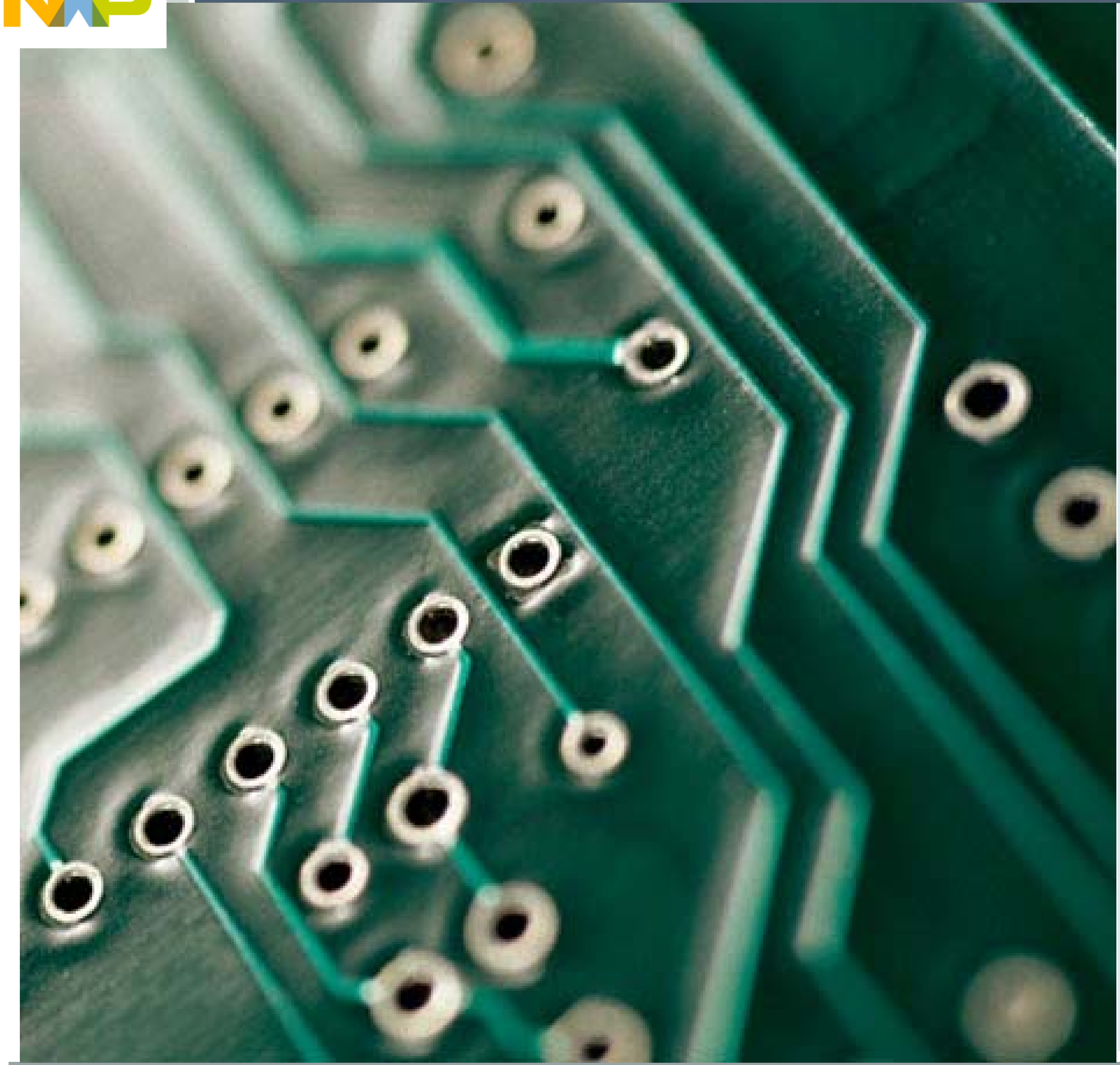


```
data rate = 953.949686 Kbps, average packet size 999
data rate = 948.100507 Kbps, average packet size 999
data rate = 905.802724 Kbps, average packet size 999
data rate = 949.133819 Kbps, average packet size 999
data rate = 877.769051 Kbps, average packet size 999
data rate = 954.472714 Kbps, average packet size 999
data rate = 943.507851 Kbps, average packet size 999
data rate = 903.264148 Kbps, average packet size 999
data rate = 953.931463 Kbps, average packet size 999
data rate = 893.048827 Kbps, average packet size 999
data rate = 951.938725 Kbps, average packet size 999
data rate = 939.755627 Kbps, average packet size 999
data rate = 920.723793 Kbps, average packet size 999
data rate = 942.547229 Kbps, average packet size 999
data rate = 943.492572 Kbps, average packet size 999
data rate = 953.842963 Kbps, average packet size 999
data rate = 954.398966 Kbps, average packet size 999
data rate = 908.092737 Kbps, average packet size 999
data rate = 954.366134 Kbps, average packet size 999
data rate = 954.187686 Kbps, average packet size 999
data rate = 954.709150 Kbps, average packet size 999
data rate = 954.419030 Kbps, average packet size 999
data rate = 919.516462 Kbps, average packet size 999
data rate = 954.399487 Kbps, average packet size 999
data rate = 948.707504 Kbps, average packet size 999
```

Adjusting the packet size

Turning the POT on the demo board changes the packet size that the UDP client sends.

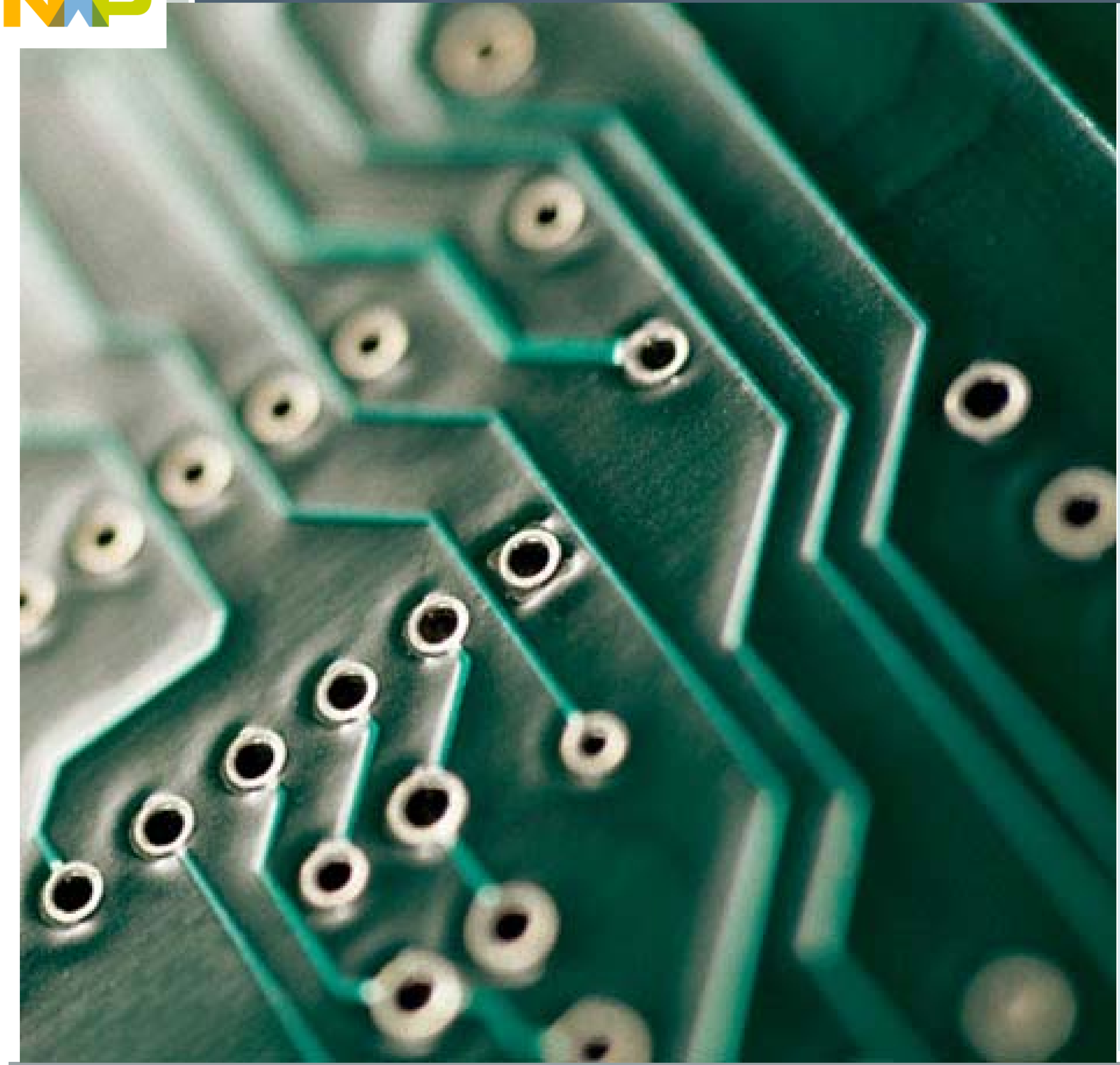
Notice the effect packet size has on data throughoutput.



LAB: UDP server

The freescale_UDP_server.c file contains a working and tested UDP server.

Unfortunately, the UDP client on the PC side is not working correctly.



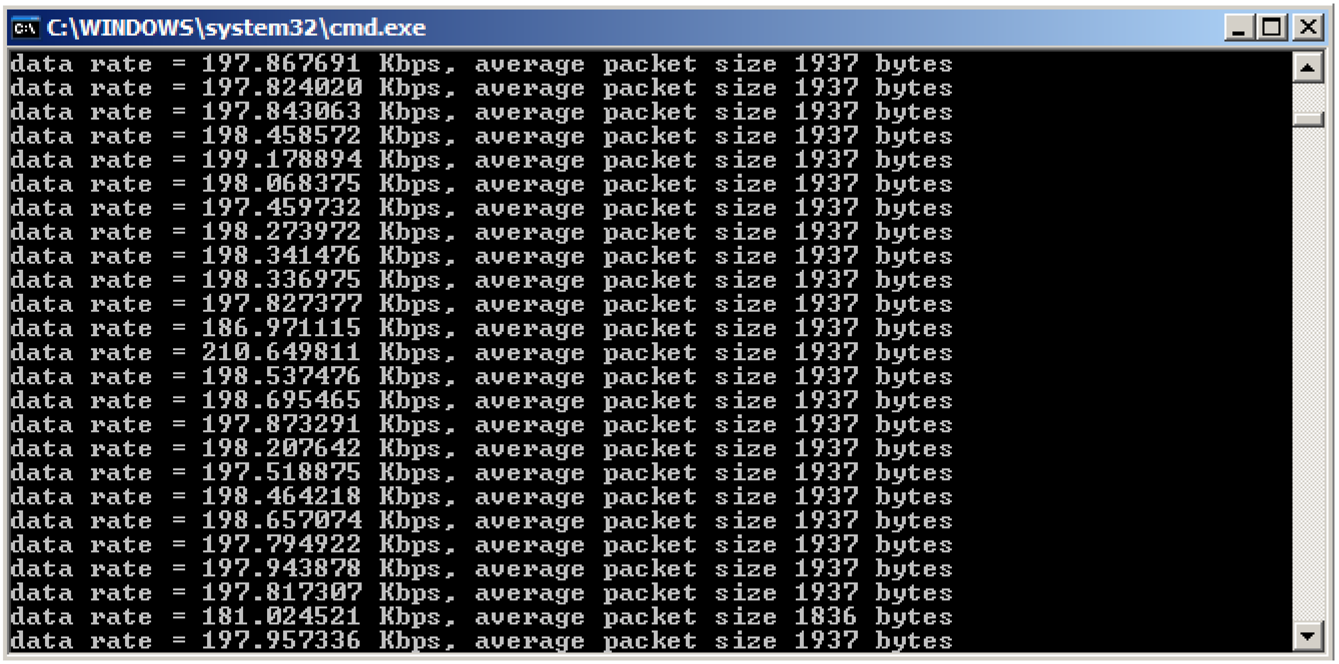
LAB: TCP client

LAB: TCP client

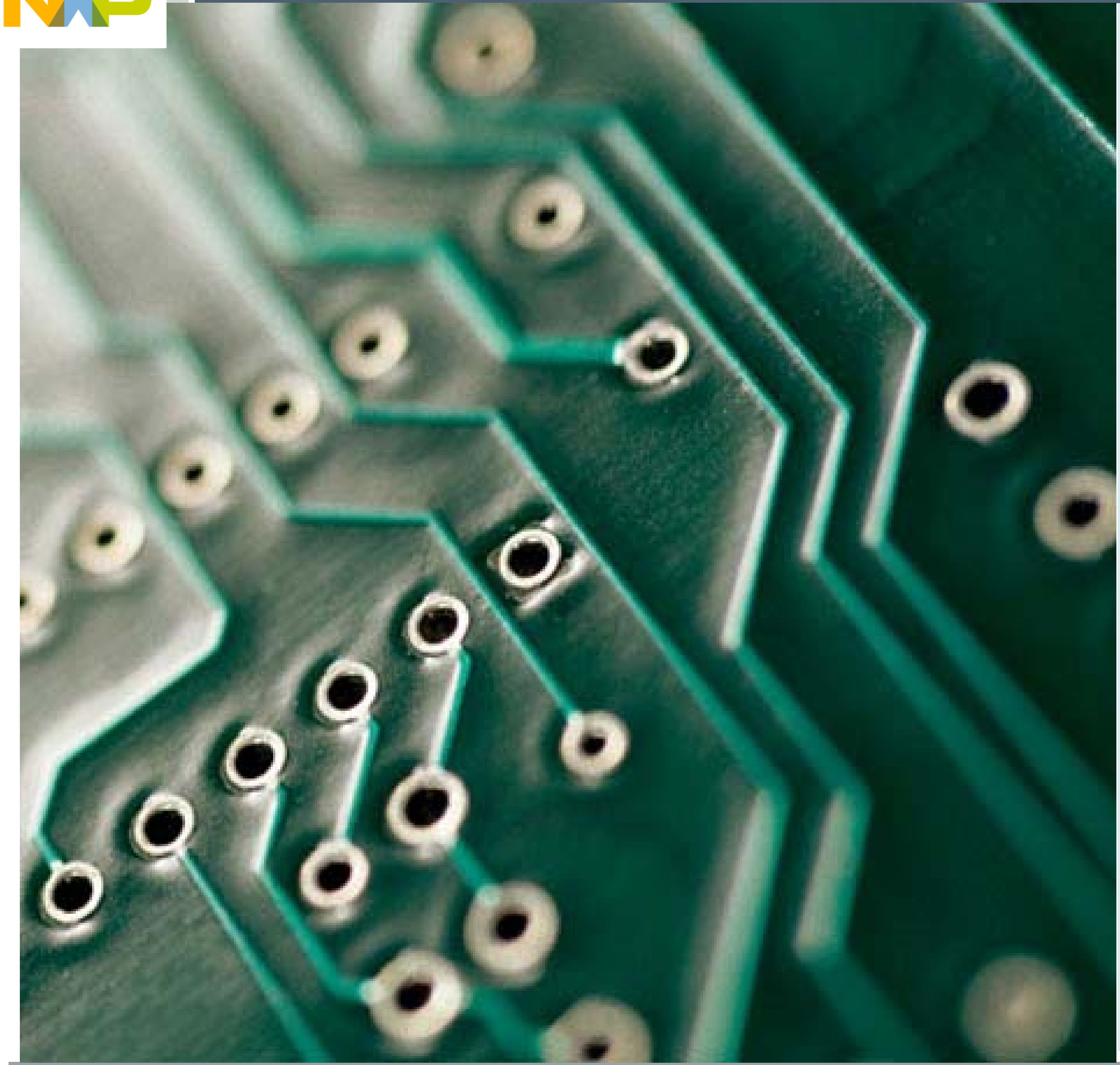
Flash the ColdFire_Lite_TCP_client target

Double click the start_TCP_server.bat

What happens when the POT is adjusted?


C:\WINDOWS\system32\cmd.exe

```
data rate = 197.867691 Kbps, average packet size 1937 bytes
data rate = 197.824020 Kbps, average packet size 1937 bytes
data rate = 197.843063 Kbps, average packet size 1937 bytes
data rate = 198.458572 Kbps, average packet size 1937 bytes
data rate = 199.178894 Kbps, average packet size 1937 bytes
data rate = 198.068375 Kbps, average packet size 1937 bytes
data rate = 197.459732 Kbps, average packet size 1937 bytes
data rate = 198.273972 Kbps, average packet size 1937 bytes
data rate = 198.341476 Kbps, average packet size 1937 bytes
data rate = 198.336975 Kbps, average packet size 1937 bytes
data rate = 197.827377 Kbps, average packet size 1937 bytes
data rate = 186.971115 Kbps, average packet size 1937 bytes
data rate = 210.649811 Kbps, average packet size 1937 bytes
data rate = 198.537476 Kbps, average packet size 1937 bytes
data rate = 198.695465 Kbps, average packet size 1937 bytes
data rate = 197.873291 Kbps, average packet size 1937 bytes
data rate = 198.207642 Kbps, average packet size 1937 bytes
data rate = 197.518875 Kbps, average packet size 1937 bytes
data rate = 198.464218 Kbps, average packet size 1937 bytes
data rate = 198.657074 Kbps, average packet size 1937 bytes
data rate = 197.794922 Kbps, average packet size 1937 bytes
data rate = 197.943878 Kbps, average packet size 1937 bytes
data rate = 197.817307 Kbps, average packet size 1937 bytes
data rate = 181.024521 Kbps, average packet size 1836 bytes
data rate = 197.957336 Kbps, average packet size 1937 bytes
```



LAB:
TCP server

LAB: TCP server

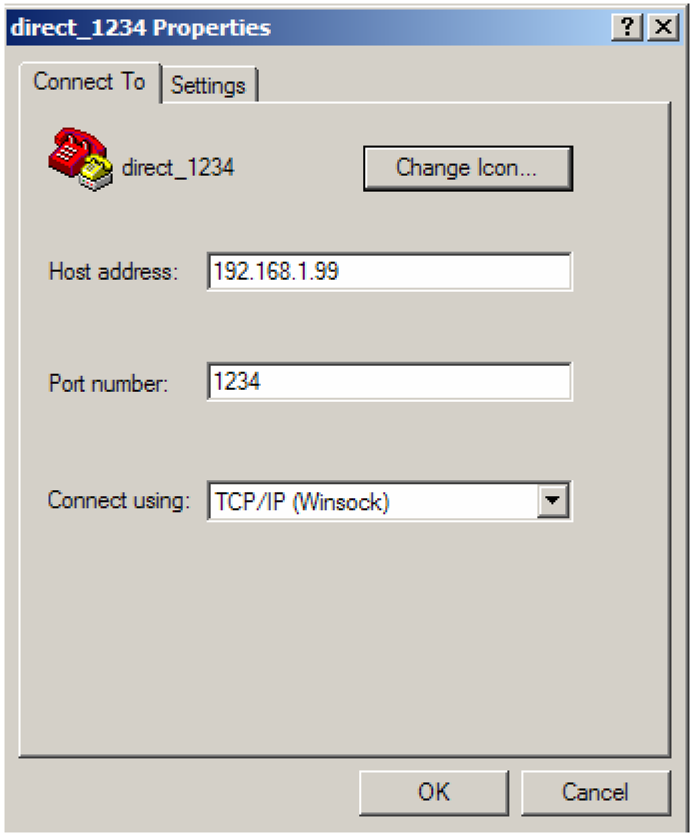
With this LAB we will measure the maximum data rate that the ColdFireLite stack can accept a TCP data stream.

Flash the ColdFire_Lite_TCP_server target

Run

Using hyperterminal transfer a test file to the board.

Hyperterminal Configuration


direct_1234 Properties

Connect To | Settings

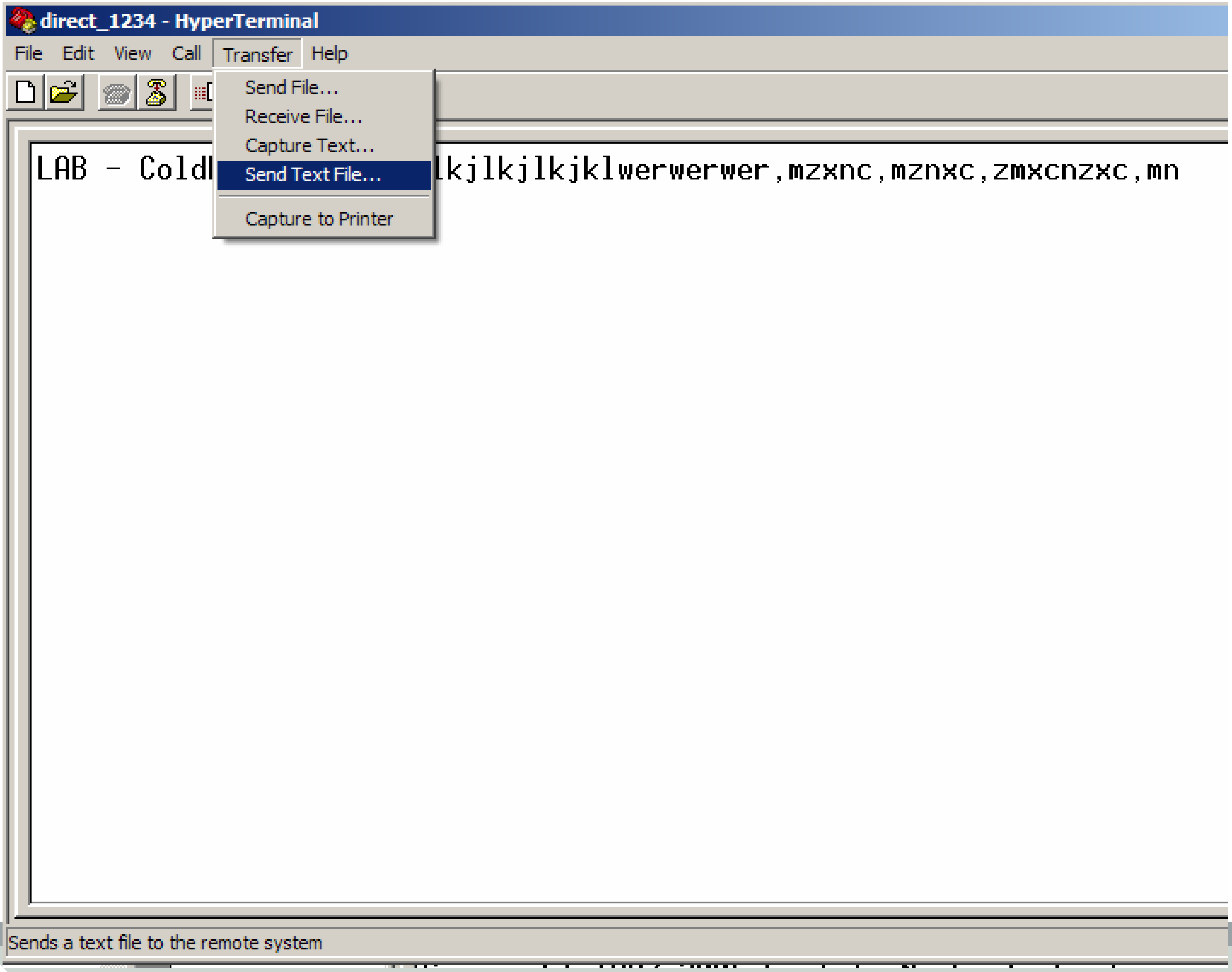
direct_1234

Host address: 192.168.1.99

Port number: 1234

Connect using: TCP/IP (Winsock)

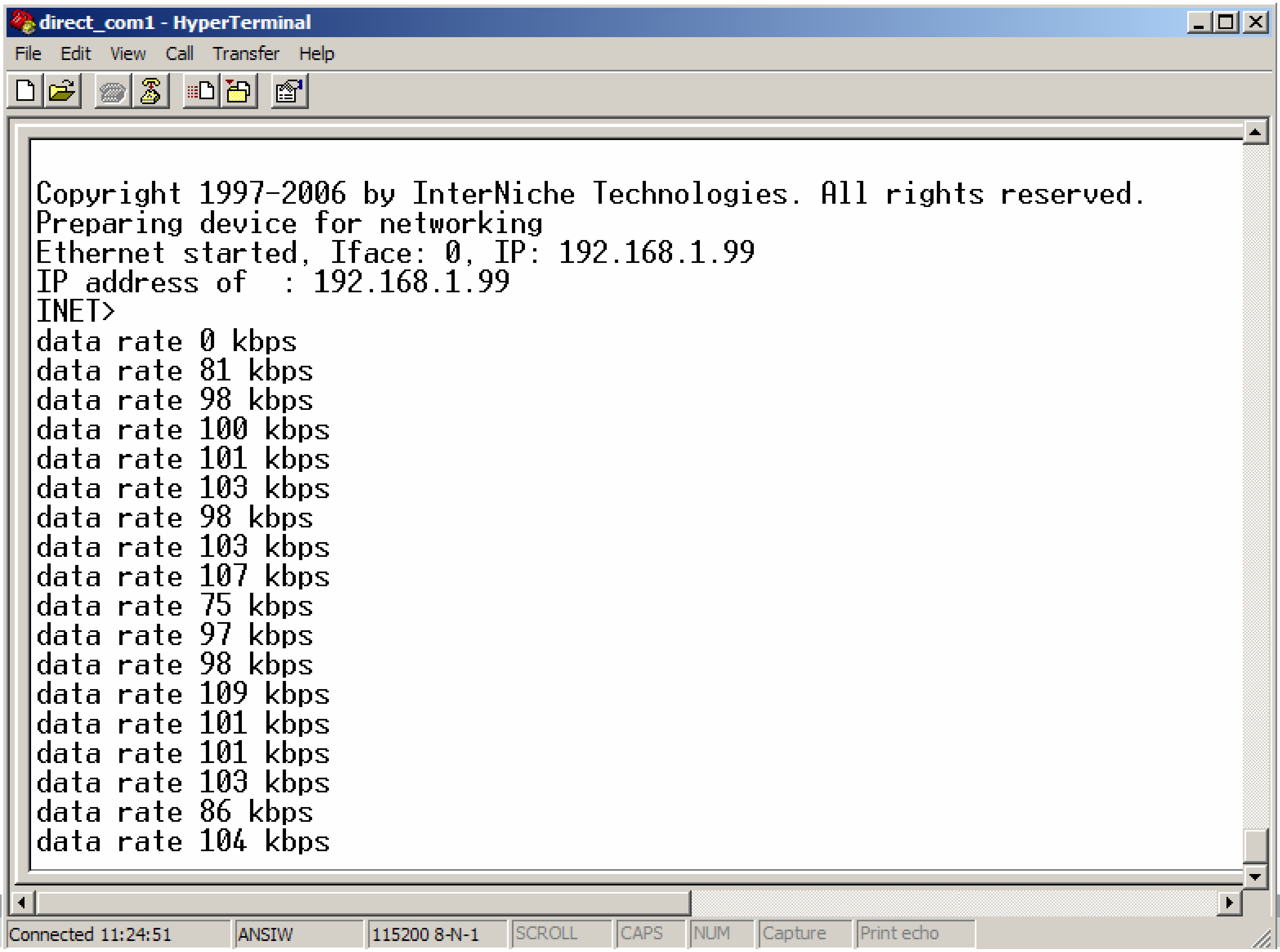
File transfer with hyperterminal



```
LAB - Coldl
lkjlkjlkjklwerwerwer,mzxnc,mznxc,zmxcnzxc,mn
```

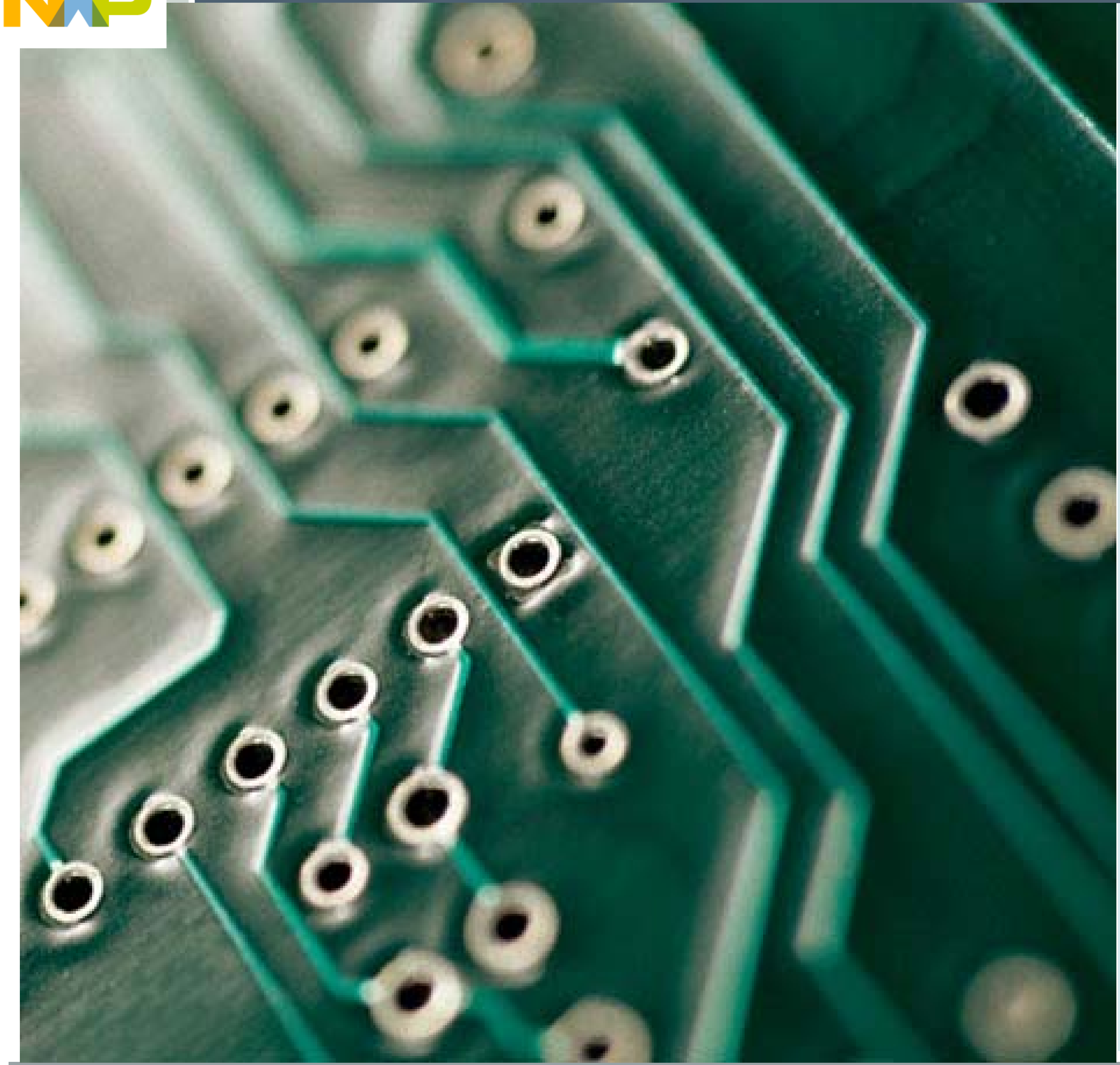
Sends a text file to the remote system

Select any file (even binary) and transfer


direct_com1 - HyperTerminal
File Edit View Call Transfer Help

```
Copyright 1997-2006 by InterNiche Technologies. All rights reserved.
Preparing device for networking
Ethernet started, Iface: 0, IP: 192.168.1.99
IP address of  : 192.168.1.99
INET>
data rate 0 kbps
data rate 81 kbps
data rate 98 kbps
data rate 100 kbps
data rate 101 kbps
data rate 103 kbps
data rate 98 kbps
data rate 103 kbps
data rate 107 kbps
data rate 75 kbps
data rate 97 kbps
data rate 98 kbps
data rate 109 kbps
data rate 101 kbps
data rate 101 kbps
data rate 103 kbps
data rate 86 kbps
data rate 104 kbps
```

Connected 11:24:51 | ANSIW | 115200 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo



LAB:
TCP serial server

Serial to Ethernet LAB

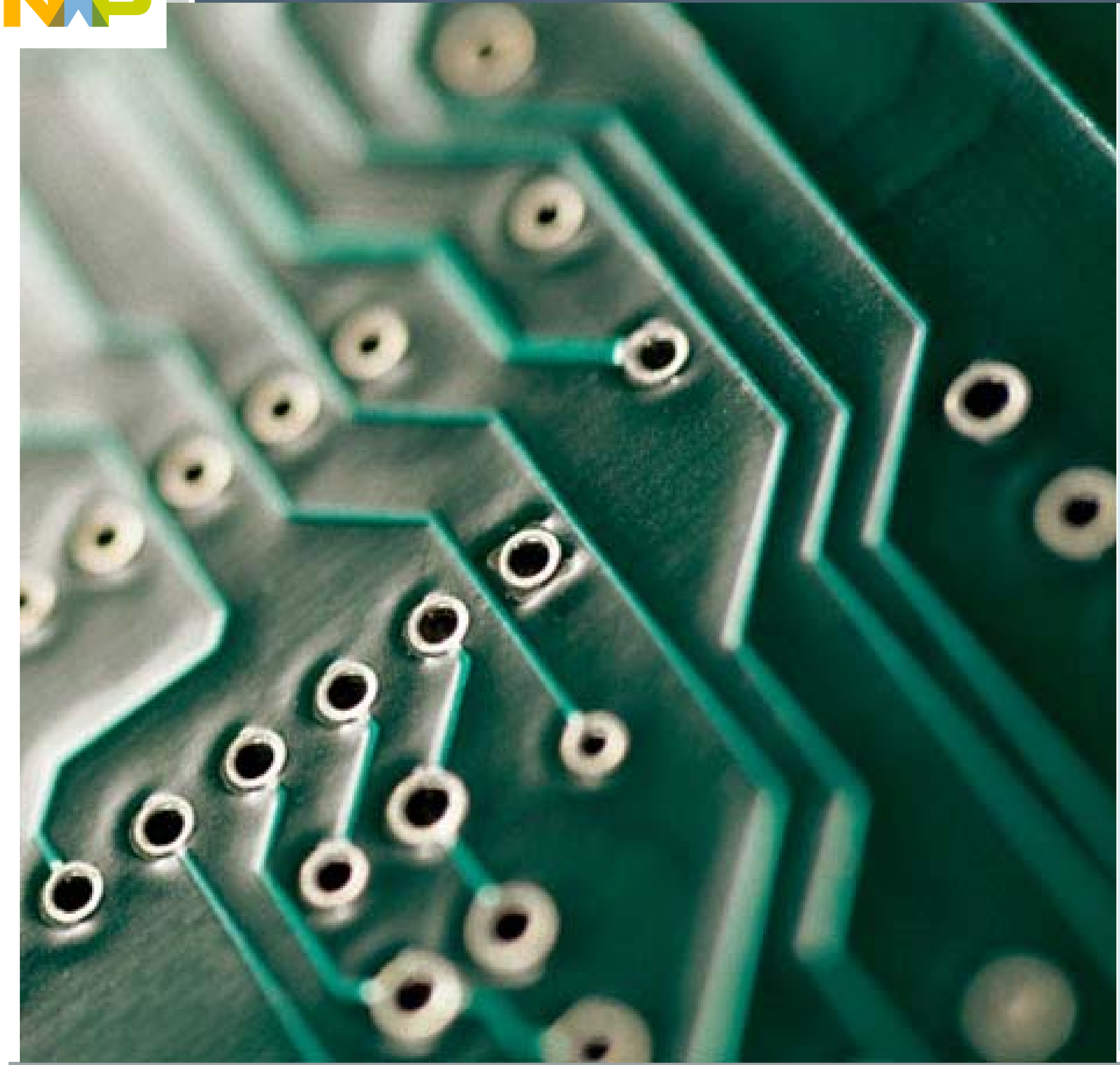
In this lab we will build a project with the Serial to Ethernet Firmware.

Flash the ColdFire_Lite_TCP_serial_server target
Run

Open 2 hyperterminal windows
COM1
TCP/IP 192.168.1.99 port 1234

Connect to the coldfire.

Anythingtyped into the TCP/IP hyperterminal appears in the serial hyperterminal, and vice-versa.



LAB: TCP serial client

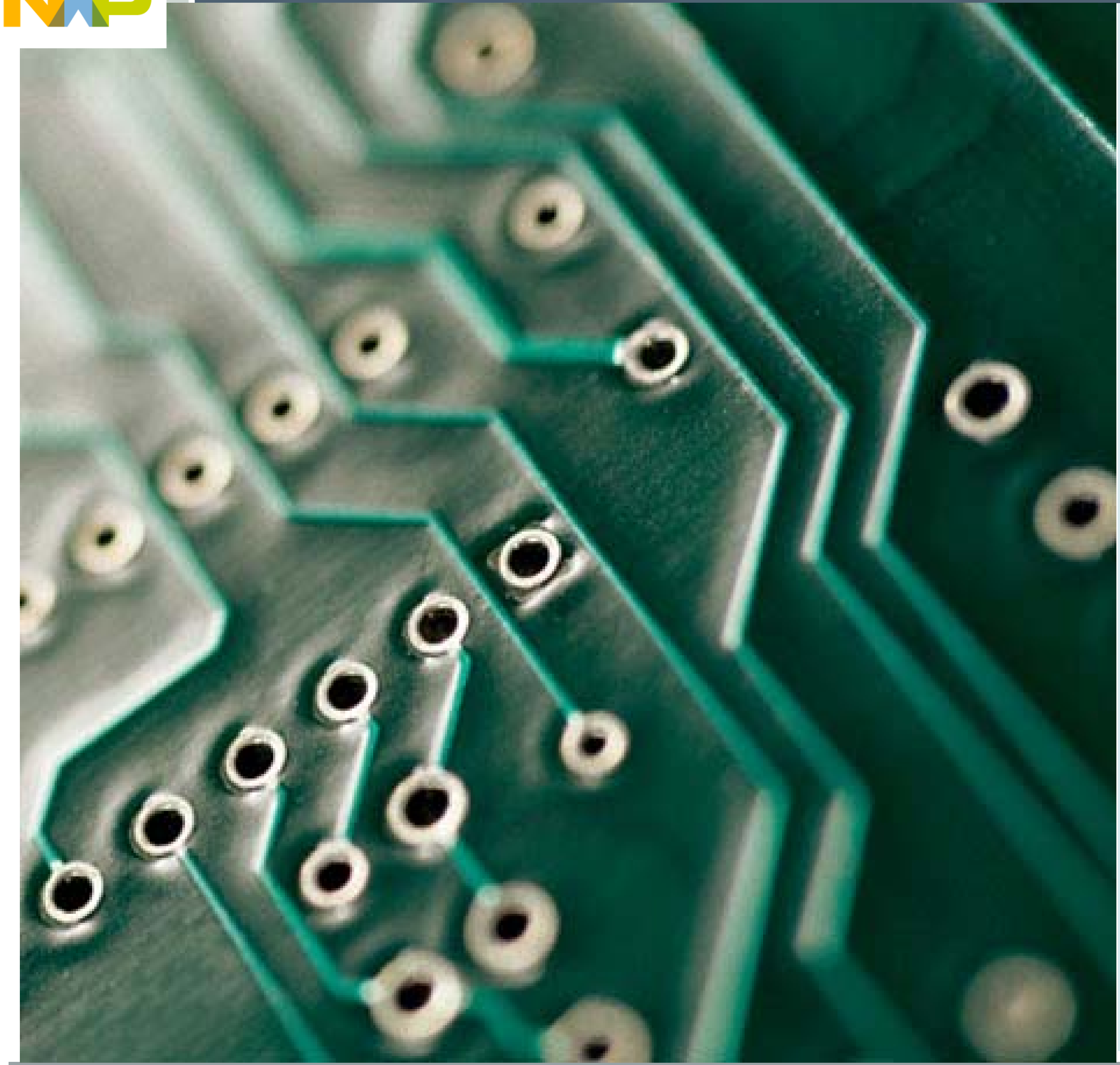
LAB: TCP serial client

For this lab, one person loads the ColdFire_Lite_TCP_serial_server target, and the other person loads the ColdFire_Lite_TCP_serial_client.

Connect the two boards via a crossover serial cable.

Each board is connected serially to a PC running hyperterminal.

When the client board is powered up, push and hold SW2.



Serial Flash Support

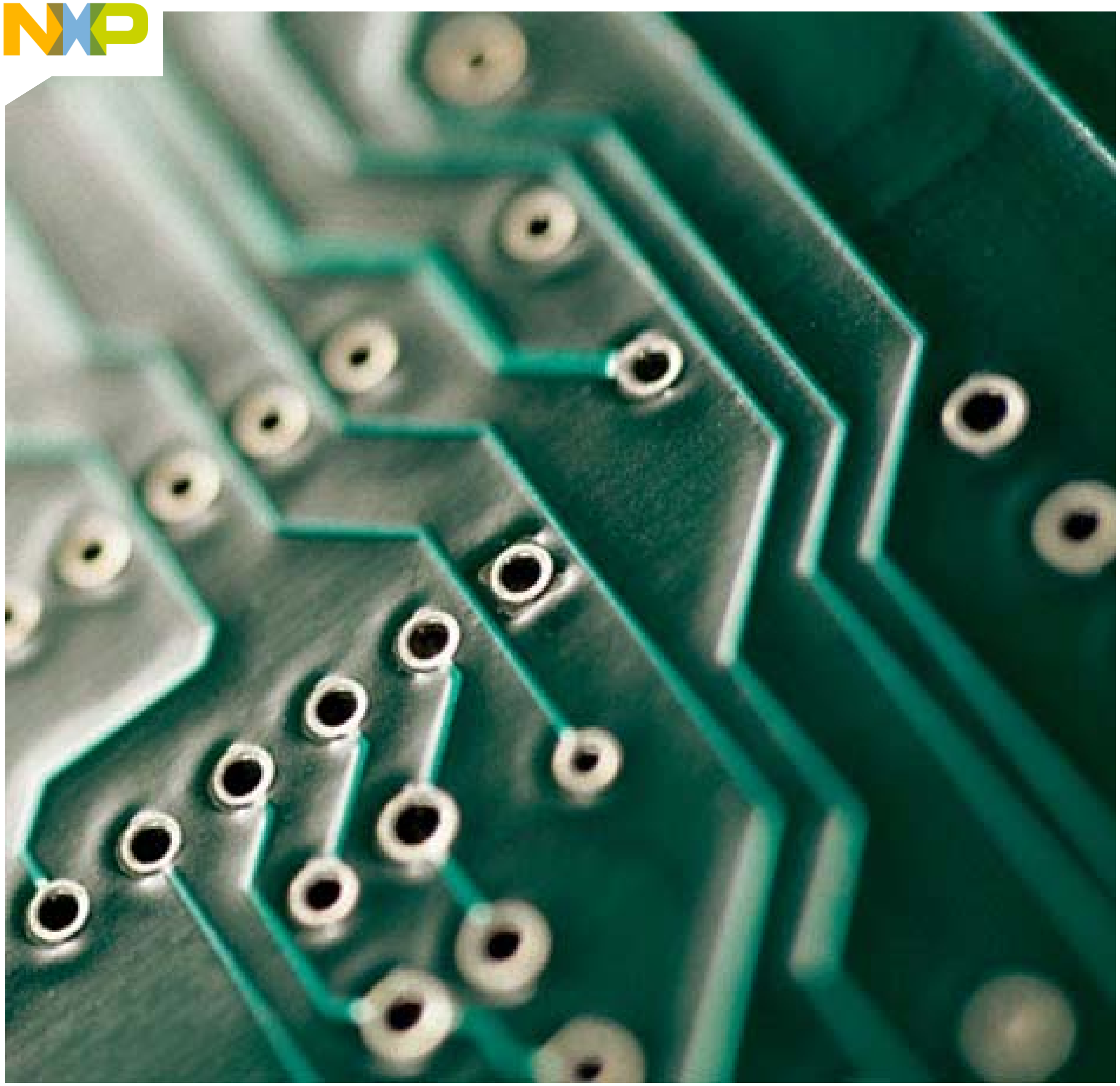
Serial Flash adds support for upto 4 meg of web pages

The serial flash is a SPI based device.

It adds the ability to store up-to 4 meg worth of web pages.

It also frees up the other 128K of internal flash for user code.

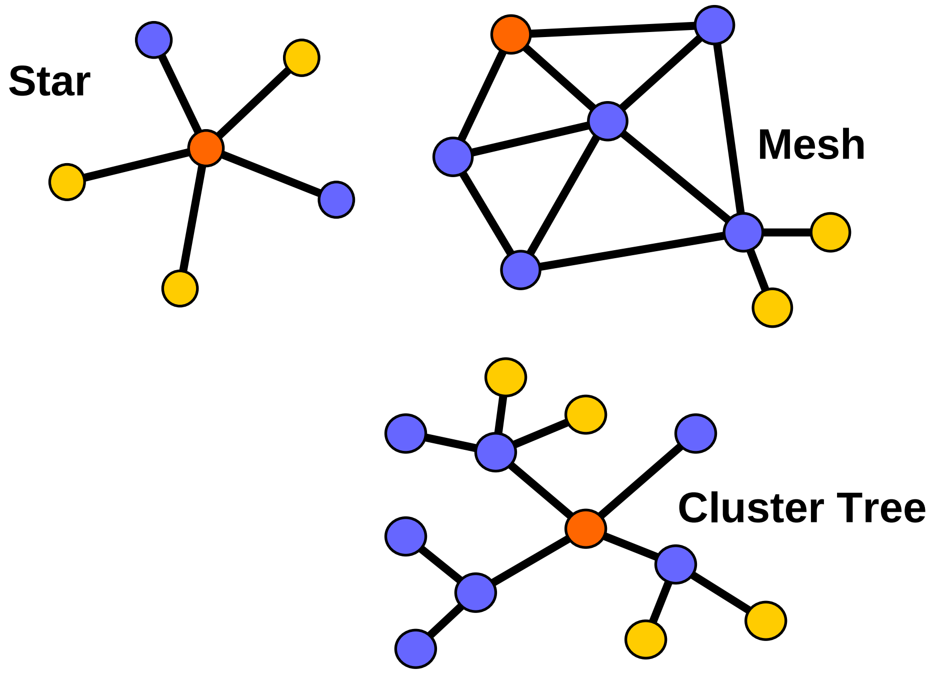
Do serial flash demo



ZigBee/802.15.4 + *ColdFire*® Ethernet = A Winning Combination

Zigbee/802.15.4 Networking

IEEE 802.15.4 Topologies



● **PAN Coordinator**

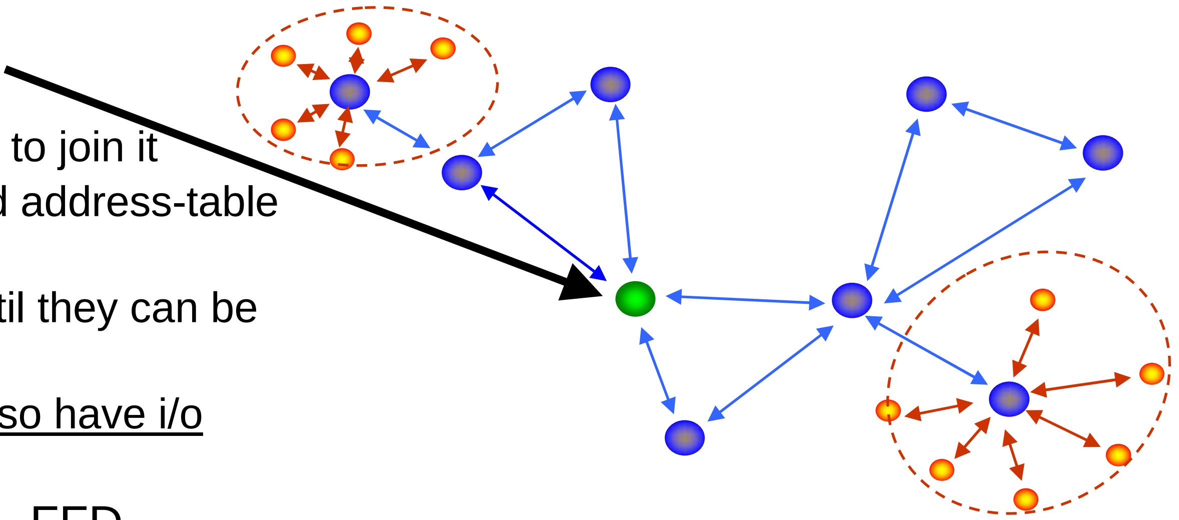
● **Full Function Device (FFD)**
- •Any topology
- •Network coordinator capable
- •Talks to any other device

● **Reduced Function Device (RFD)**
- •Limited to being leaf devices
- •Cannot become a network coordinator
- •Talks only to a network coordinator
- •Very simple implementation

Network Pieces –PAN Coordinator

PAN Coordinator

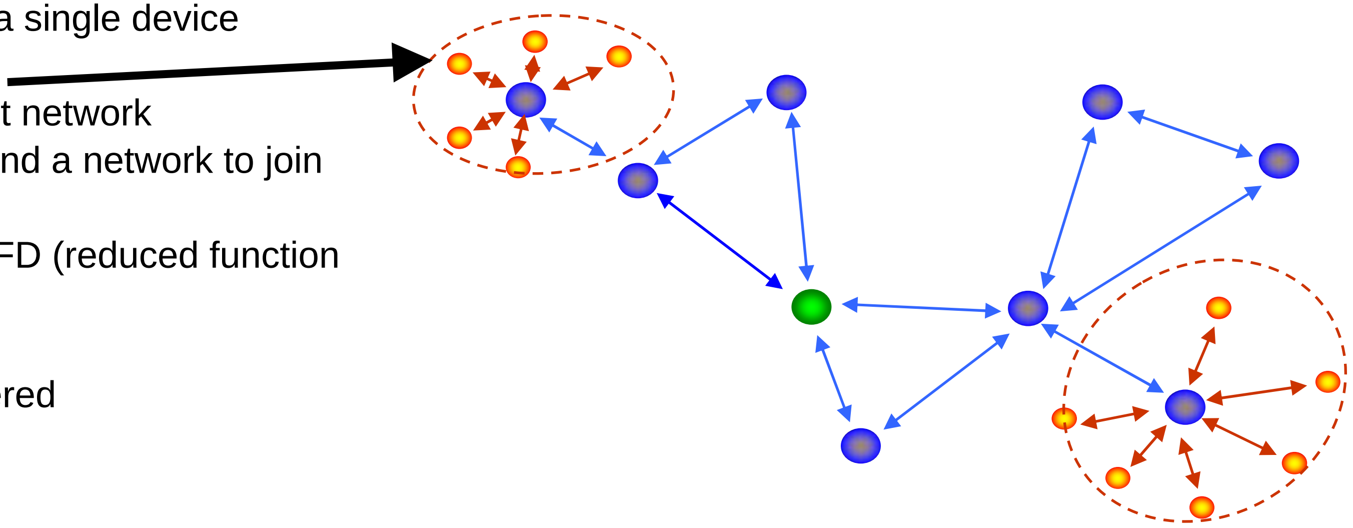
- “owns” the network
    - Starts it
    - Allows other devices to join it
    - Provides binding and address-table services
    - Saves messages until they can be delivered
    - And more… could also have i/o capability
- A “full-function device” – FFD
- Mains powered



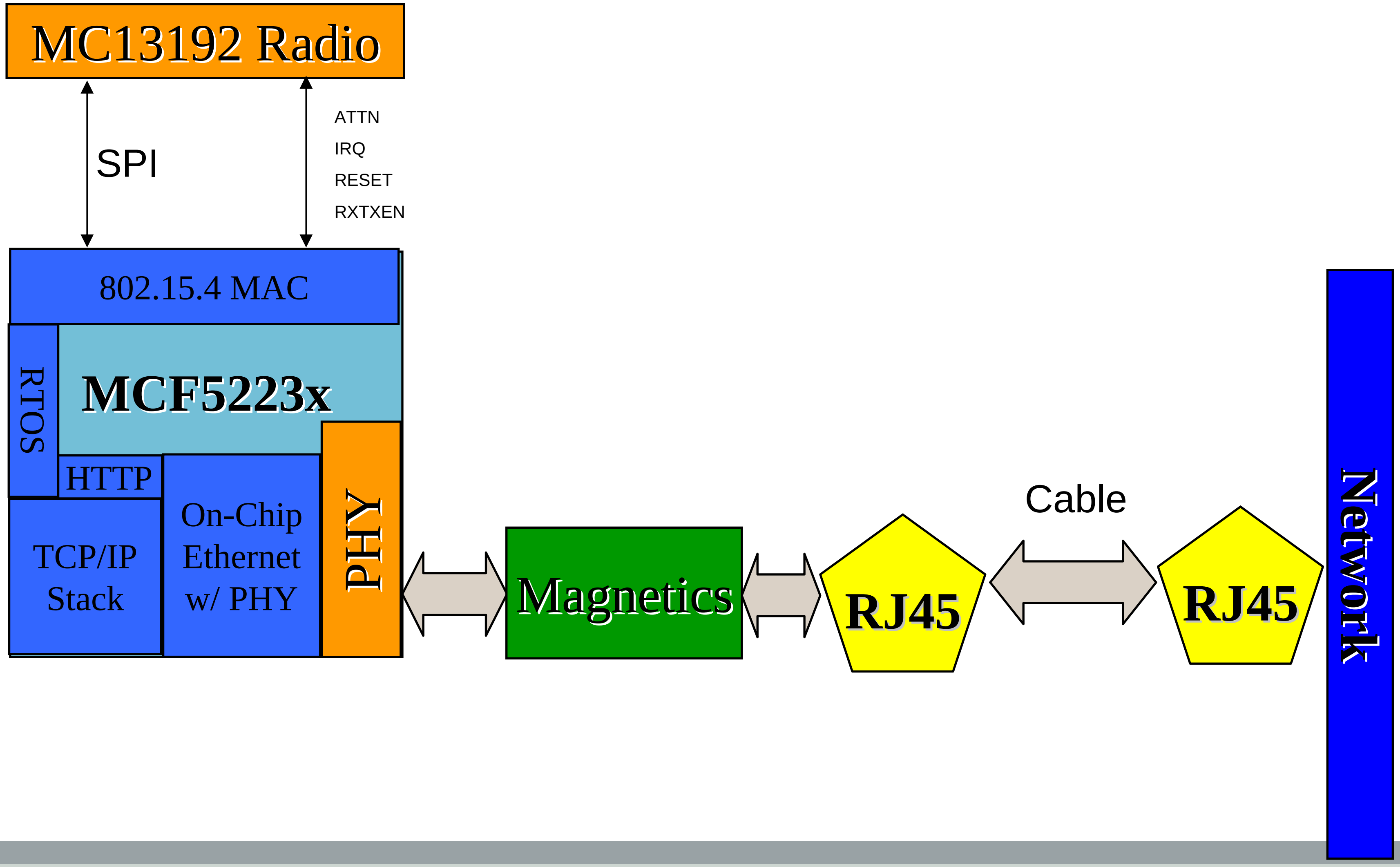
Network Pieces – End Device

End Device

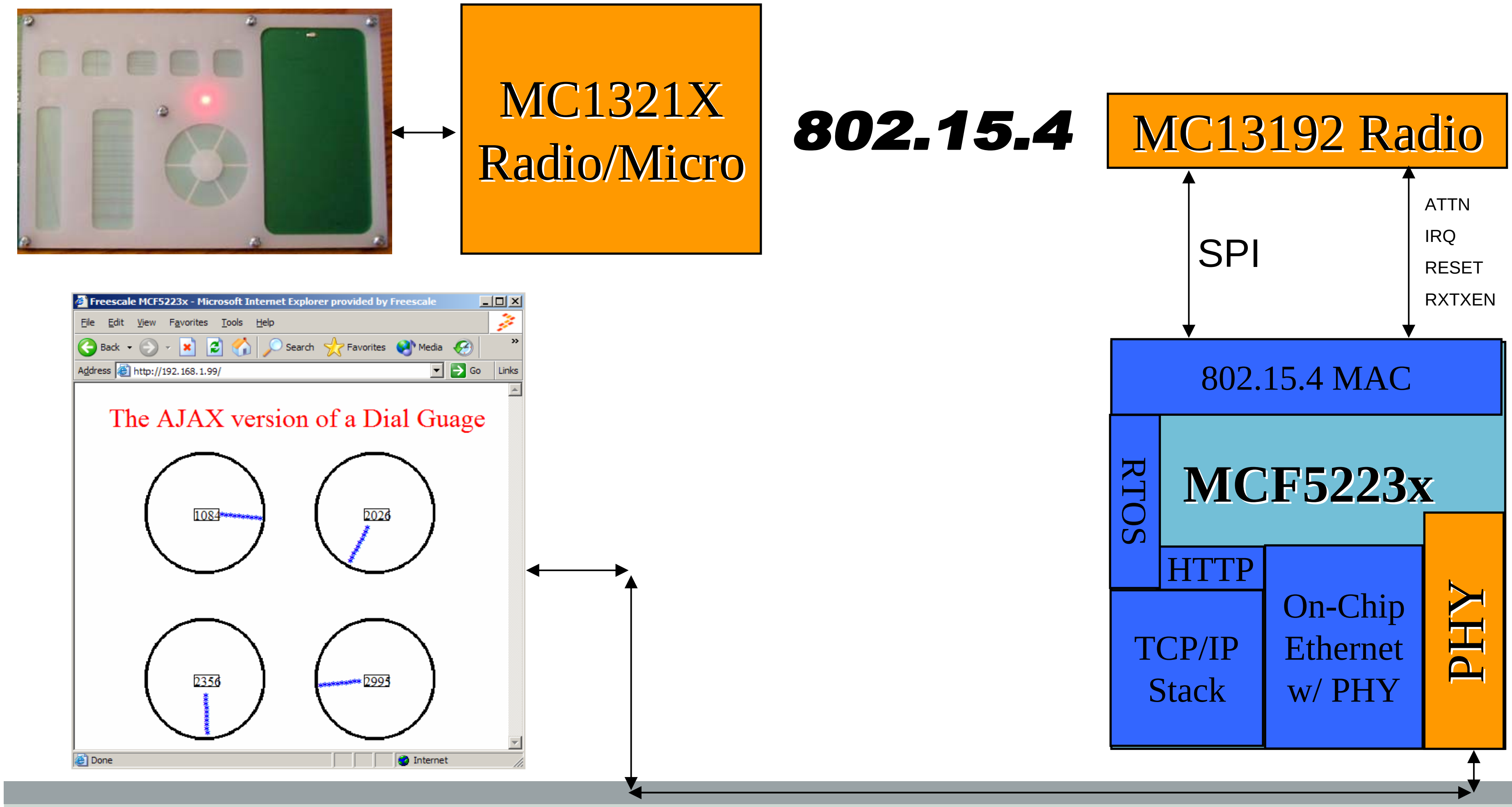
- Communicates with a single device
- Does not own or start network
  - Scans to find a network to join
- Can be an FFD or RFD (reduced function device)
- Usually battery powered



TCP/IP stack merged with 802.15.4



System Diagram


MC1321X
Radio/Micro
802.15.4
MC13192 Radio
SPI
ATTN
IRQ
RESET
RXTXEN
802.15.4 MAC
RTOS
MCF5223x
HTTP
TCP/IP Stack
On-Chip Ethernet w/ PHY
PHY
Freescale MCF5223x - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.1.99/
Go Links
The AJAX version of a Dial Guage
1084
2026
2356
2995
Done
Internet

Idd

The sensors spend most of their time in hybernate mode.

In hybernate mode, each sensor only draws 4µA.

Each sensor wakes up every 5 seconds as a heartbeat, using the RTI.

If the sensor detects a trigger, it wakes up immediately to send its data.

Assuming less then one trigger every 5 seconds, each sensor should get a battery life of over 3 years using 2 AA’s.

The coordinator is always powered up.

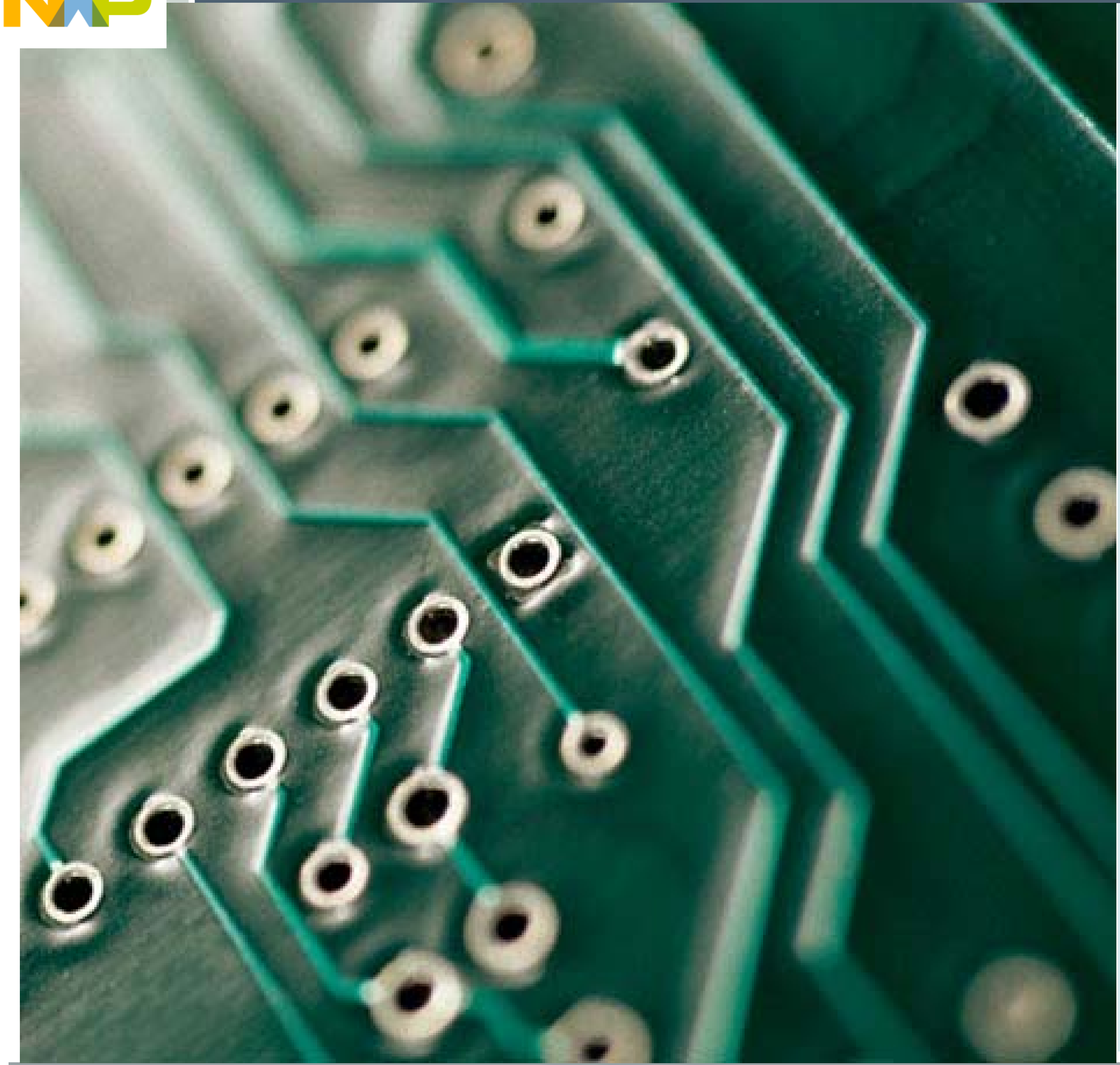
The Web Interface

The web server provides a easy method of connecting external embedded systems over serial.

The external embedded system can send data to the web server using the VAR command.

The web server can send data over serial to the embedded system using forms.

This provides a simple mechanism for getting your embedded system on the web.



The ColdFire Lite Folder

The ColdFireLite folder

The ColdFireLite folder contains the deliverables from Interniche.
The Interniche stack supports;

- A simple RTOS
- IP protocol with ICMP and ARP
- UDP protocol
- TCP protocol
- A simple mini-socket API for TCP
- A DHCP client
- A DNS client
- A PING client
- A serial console with configurable menus
- A TFTP server and TFTP client
- A RAM based file system

Allports folder

Allports folder

- Allports.c - netmain_init()
- Timeouts.c - inet_timers() and check_interval_timers()
- Tk_misc.c - Contains the “console” task

These files are used in the boot process.

headers folder

headers folder

- There are many files in this folder.
- We will concentrate on ipport.h and osport.h

osport.h

- Contains defines associated with the RTOS.
- Contains the standard stack sizes for application tasks.

Ipport.h

- Contains defines associated with the TCP/IP stack.
- Contains switches to enable/disable features of the TCP/IP stack.

Mcf_specific folder

Cksum.s

- RFC1071 assembly language checksum routine.

Ifec.c

- Fast Ethernet Controller driver.

Iutils.c

- Low level serial routines

Tk_utils.s

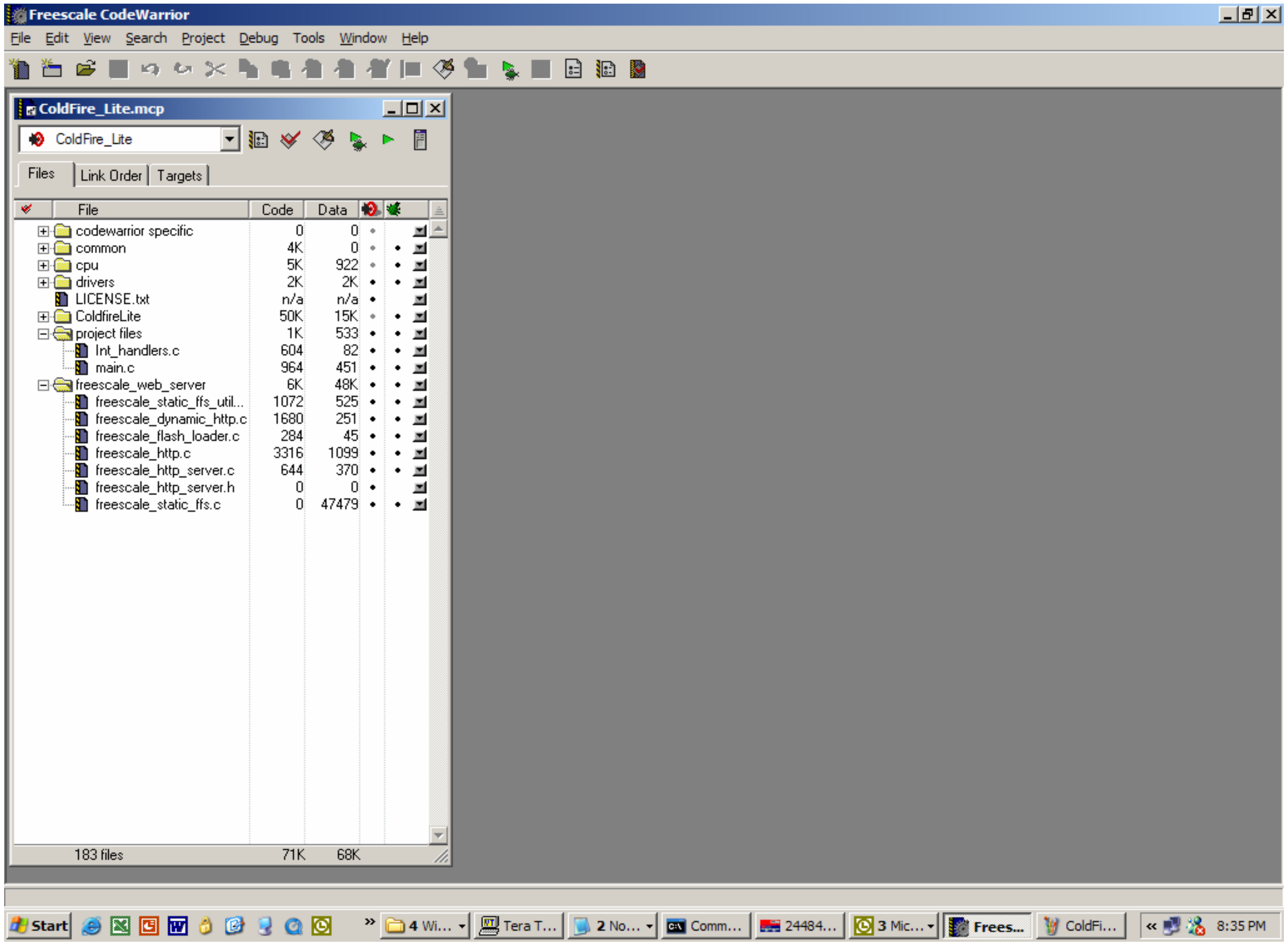
- Contains defines associated with the TCP/IP stack.
- Contains switches to enable/disable features of the TCP/IP stack.

Mcf_specific folder

Ipport.h componant selection

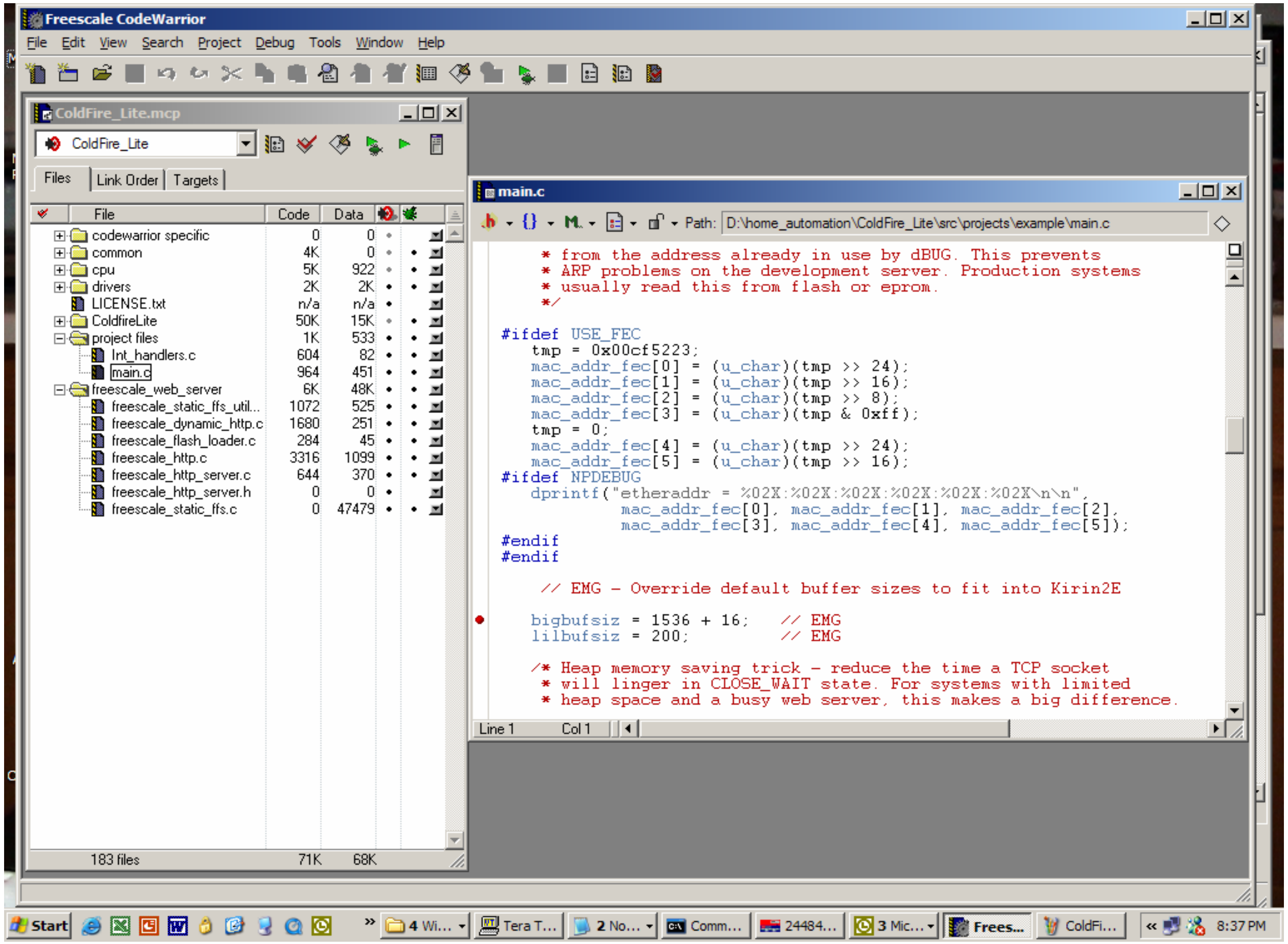
```
/*
 * Option macros to trade off features for size. Do not enable options
 * for modules you don't have or your link will get unresolved
 * externals.
 */

#define INCLUDE_ARP              1  /* use Ethernet ARP */
#define FULL_ICMP                1  /* use all ICMP || ping only */
#define OMIT_IPV4                1  /* not IPV4, use with MINI_IP */
#define MINI_IP                  1   /* Use Nichelite mini-IP layer */
#define MINI_TCP                 1   /* Use Nichelite mini-TCP layer */
#define MINI_PING                1   /* Build Light Weight Ping App for Niche Lite */
#define BSDISH_RECV              1   /* Include a BSD recv()-like routine with mini_tcp */
#define BSDISH_SEND              1   /* Include a BSD send()-like routine with mini_tcp */
#define NB_CONNECT               1  /* support Non-Blocking connects (TCP, PPP, et al) */
#define MUTE_WARNS               1  /* gen extra code to suppress compiler warnings */
#define IN_MENUS                 1  /* support for InterNiche menu system */
#define NET_STATS                1  /* include statistics printfs */
#define QUEUE_CHECKING           1  /* include code to check critical queues */
#define INICHE_TASKS             1  /* InterNiche multitasking system */
#define MEM_BLOCKS               1  /* list memory heap stats */
// EMG #define TFTP_CLIENT      1  /* include TFTP client code */
// EMG #define TFTP_SERVER      1  /* include TFTP server code */
// EMG #define DNS_CLIENT       1  /* include DNS client code */
#define INICHE_TIMERS            1  /* Provide Interval timers */

// EMG - To enable DHCP, uncomment the line below
#define DHCP_CLIENT              1  /* include DHCP client code */

// EMG #define INCLUDE_NVPARMS  1  /* non-volatile (NV) parameters logic */
#define NPDEBUG                  1  /* turn on debugging dprintf()s */
// EMG #define VFS_FILES        1  /* include Virtual File System */
// EMG #define USE_MEMDEV       1   /* Psuedo VFS files mem and null */
#define NATIVE_PRINTF            1   /* use target build environment's printf function */
#define NATIVE_SPRINTF           1   /* use target build environment's printf function */
#define PRINTF_STDARG            1   /* build ...printf() using stdarg.h */
#define TK_STDIN_DEVICE          1   /* Include stdin (uart) console code */
#define BLOCKING_APPS            1   /* applications block rather than poll */
#define INCLUDE_TCP              1  /* this link will include NetPort TCP w/MIB */

/**** end of option list ***/
```

Open The directory Containing MAIN.C



File	Code	Data		
codewarrior specific	0	0	•	
common	4K	0	•	•
cpu	5K	922	•	•
drivers	2K	2K	•	•
LICENSE.txt	n/a	n/a	•	
ColdfireLite	50K	15K	•	•
project files	1K	533	•	•
Int_handlers.c	604	82	•	•
main.c	964	451	•	•
freescale_web_server	6K	48K	•	•
freescale_static_ffs_util...	1072	525	•	•
freescale_dynamic_http.c	1680	251	•	•
freescale_flash_loader.c	284	45	•	•
freescale_http.c	3316	1099	•	•
freescale_http_server.c	644	370	•	•
freescale_http_server.h	0	0	•	
freescale_static_ffs.c	0	47479	•	•
183 files	71K	68K		

Open main.c



Freescale CodeWarrior

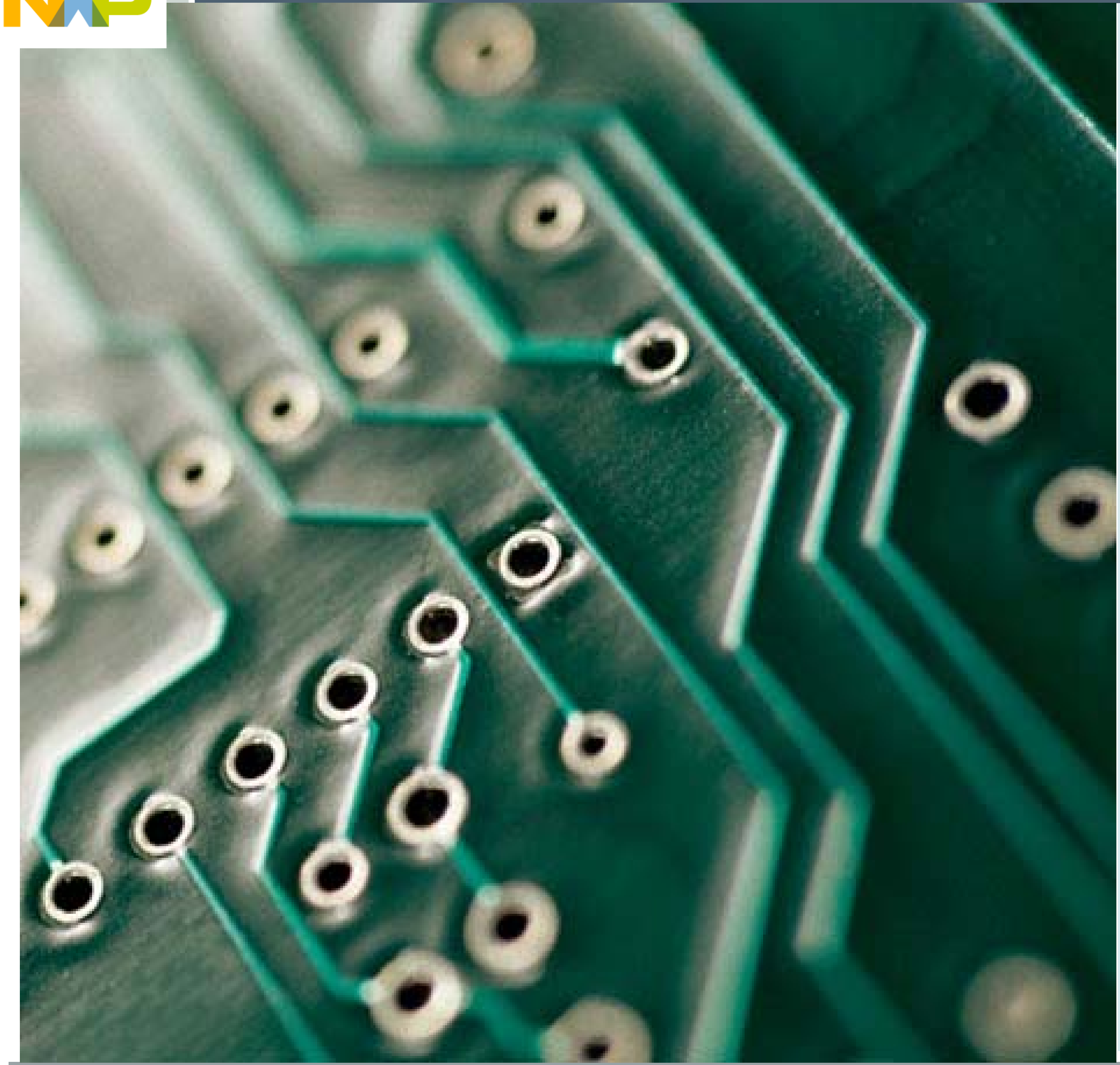
ColdFire_Lite.mcp

File	Code	Data
codewarrior specific	0	0
common	4K	0
cpu	5K	922
drivers	2K	2K
LICENSE.txt	n/a	n/a
ColdfireLite	50K	15K
project files	1K	533
Int_handlers.c	604	82
main.c	964	451
freescale_web_server	6K	48K
freescale_static_ffs_util...	1072	525
freescale_dynamic_http.c	1680	251
freescale_flash_loader.c	284	45
freescale_http.c	3316	1099
freescale_http_server.c	644	370
freescale_http_server.h	0	0
freescale_static_ffs.c	0	47479
183 files	71K	68K

main.c — Path: D:\home_automation\ColdFire_Lite\src\projects\example\main.c

```
     * from the address already in use by dBUG. This prevents
     * ARP problems on the development server. Production systems
     * usually read this from flash or eprom.
     */

#ifdef USE_FEC
    tmp = 0x00cf5223;
    mac_addr_fec[0] = (u_char)(tmp >> 24);
    mac_addr_fec[1] = (u_char)(tmp >> 16);
    mac_addr_fec[2] = (u_char)(tmp >> 8);
    mac_addr_fec[3] = (u_char)(tmp & 0xff);
    tmp = 0;
    mac_addr_fec[4] = (u_char)(tmp >> 24);
    mac_addr_fec[5] = (u_char)(tmp >> 16);
#ifdef NPDEBUG
    dprintf("etheraddr = %02X:%02X:%02X:%02X:%02X:%02X\n\n",
            mac_addr_fec[0], mac_addr_fec[1], mac_addr_fec[2],
            mac_addr_fec[3], mac_addr_fec[4], mac_addr_fec[5]);
#endif
#endif

     // EMG - Override default buffer sizes to fit into Kirin2E

    bigbufsiz = 1536 + 16;    // EMG
    lilbufsiz = 200;          // EMG

    /* Heap memory saving trick - reduce the time a TCP socket
     * will linger in CLOSE_WAIT state. For systems with limited
     * heap space and a busy web server, this makes a big difference.
```

Line 1 Col 1



HTTP/HTML/AJAX Overview

(And ColdFire TCP/IP Lite

ColdFire_TCP/IP_Lite Stack

The ***ColdFire***_TCP/IP_Lite stack includes:

A Mini-Sockets TCP API.

A TFTP (Trivial File Transfer protocol) server.

A DHCP (Dynamic Host Configuration protocol) client.

Zero-copy sockets for performance.

Less then 40K of program space.

Mini-Sockets TCP API

The mini-Sockets API is designed to be as close as possible to the BSD Sockets API and still allow a small footprint. The primary differences are that passive connections are accomplished with a single call, m_listen(), rather than the BSD bind()-listen()-accept() sequence, and the BSD select() call is replaced with a callback mechanism.

BSD = **B**erkeley **S**oftware **D**istribution

Mini-Socket Interface Compared to BSD Sockets

Mini-Sockets	BSD Sockets
m_socket()	socket()
m_connect()	connect()

m_recv() and/or m_send()
- or -
tcp_send() and/or tcp_recv() - (zero-copy I/O)
	recv() and/or send()

m_close()	close();

For server applications:

Mini-Sockets	BSD Sockets
(n/a - merged with listen)	socket()
(n/a - merged with listen)	bind()
m_listen()	listen()
(n/a - handled via callback)	accept()

m_recv() and/or m_send()
- or -
tcp_send() and/or tcp_recv() - (zero-copy I/O)
	recv() and/or send()

m_close()	close();

A Simple Server Using Mini-Sockets

Creating a Listening Socket

```
// Init a socket structure with our Port Number
emg_http_sin.sin_addr.s_addr                    = (INADDR_ANY);
emg_http_sin.sin_port           = (PORT_NUMBER);

 emg_http_server_socket       = m_listen(&emg_http_sin, freescale_http_cmdcb, &e);
```

Accepting a Connection

```
switch(code)
{
        // socket open complete
        case M_OPENOK:
                msring_add(&emg_http_msring, so);
                break;
}
```

Receiving TCP data

```
length = m_recv( freescale_http_sessions[session].socket, (char *)buffer, RECV_BUFFER_SIZE );
```

Sending TCP data

```
bytes_sent = m_send( freescale_http_sessions[session].socket, data, length );
```

Closing the Socket

```
j = m_close( so );
```

A Simple Client Using Mini-Sockets

Creating a Socket

```
M_SOCK Socket = m_socket();
```

Connecting to a Server

```
int m_connect(M_SOCK socket, struct sockaddr_in * sin, M_CALLBACK(name));
// m_connect is blocking until a connection completes.
// If the socket is configured for non-blocking, then the callback funtion is used to indicate when the connection is established.
```

Receiving TCP data

```
length = m_recv( freescale_http_sessions[session].socket, (char *)buffer, RECV_BUFFER_SIZE );
```

Sending TCP data

```
bytes_sent = m_send( freescale_http_sessions[session].socket, data, length );
```

Closing the Socket

```
j = m_close( so );
```

Freescale Web Server

HTTP1.0 compliant server with connection persistance and multiple sessions (HTTP1.1 will be available in future revisions).

GET and POST elements supported.

Dynamic HTML support with replace and conditional tokens.

Serial interface support for Dynamic HTML variables.

Provides run time and compile time flash file systems.

Long file name support with subdirectories.

‘DIR’ command supported on serial interface.

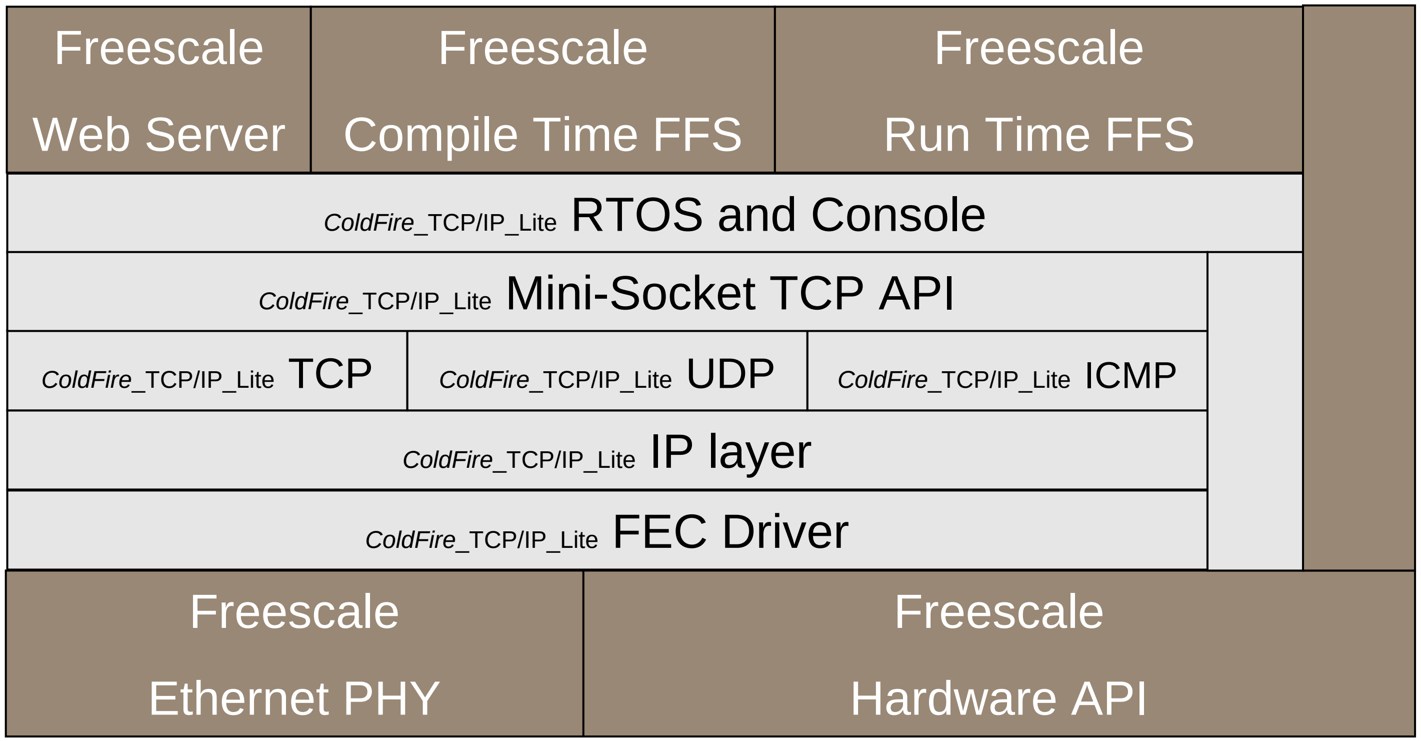
PC utilities for compressing compile time and run time downloadable images of multi-page web pages.

PC utility for downloading run time downloadable web page image through port 80 (to get through firewalls).

32 byte ascii key for web page download security.

It’s Free for use on *ColdFire*® processors!!!

Software Model


Freescale
Web Server
Freescale
Compile Time FFS
Freescale
Run Time FFS
ColdFire_TCP/IP_Lite RTOS and Console
ColdFire_TCP/IP_Lite Mini-Socket TCP API
ColdFire_TCP/IP_Lite TCP
ColdFire_TCP/IP_Lite UDP
ColdFire_TCP/IP_Lite ICMP
ColdFire_TCP/IP_Lite IP layer
ColdFire_TCP/IP_Lite FEC Driver
Freescale
Ethernet PHY
Freescale
Hardware API

FFS = Flash File System

Web Servers

Web Servers implement the HyperText Transfer Protocol (HTTP) to send web pages from a server to a client.

The Web Server contains the content, the Web Browser Displays the content.

For these labs, the Web Browser used will be the Internet Explorer.

HTTP - An Overview

HTTP – HyperText Transport Protocol.
HTTP – Is used to transfer HTML/Web Pages on the web.
From RFC1945:

> *The HTTP protocol is based on a request/response paradigm. A client establishes a connection with a server and sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content. The server responds with a status line, including the messages protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible body content.*

Generally HTTP uses TCP/IP port 80.

There are two versions of HTTP, 1.0 and 1.1.

HTTP1.0 is defined by RFC1945.

HTTP Protocol Exchange

The client starts an exchange using one of two Methods:

GET method – Request the server to send a file

POST method – Sends a file to the server

- The method is followed by a list of Request Header Fields

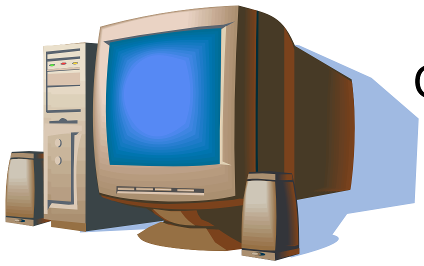
The server responds with a response message:

The first line of the message is the status line.

- Sample Status line HTTP/1.0 200 OK
  - Status code 2xx means success
  - Status code 4xx means error

The status line is followed by a series of entity header fields separated by carriage return/line feeds.

HTTP Request / Response



HTTP Request

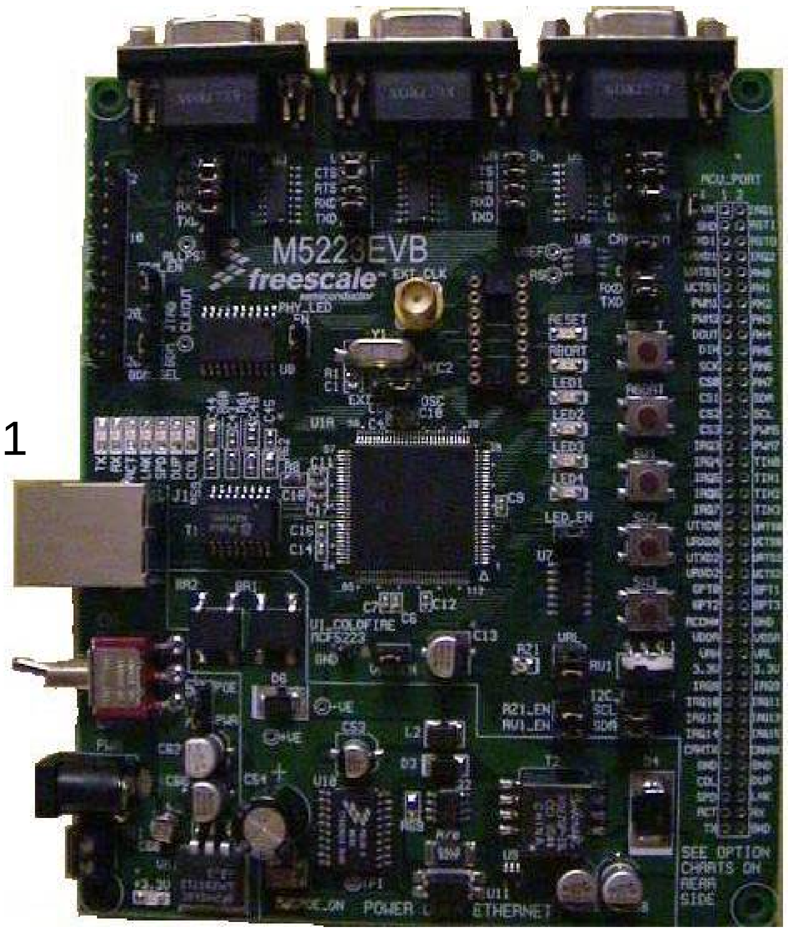
$\longrightarrow$

GET /filename.htm HTTP/1.1

HTTP Response

$\longleftarrow$

HTTP/1.1 200 OK



The Client (Browser) HTTP Request

```
GET /filename.htm HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword
Accept-language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozzilla/4.0 (compatable; MSIE 6.0; Windows NT 5.1 )
Host: www.msn.com
Connection: Keep-Alive
```

The above text is sent to the server on TCP/IP port 80

It asks the server to respond with the contents of filename.htm

It tells the server that it supports the HTTP1.1 standard

It tells the server that the client supports: gif, x-xbitmaps, jpeg, and pjpeg images

It tells the server that it supports msword documents

It tells the server that the language is English, and that the gzip and deflate decompression algorithm’s are available

It tells the server that the browser is running IE6.0 on a Windows machine

Finally it tells the server NOT to close the connection after the file is sent

Keep-Alive

By default, after the server sends the file to the client, it closes the TCP/IP connection.

The Keep-Alive request header field tells the server NOT to close the TCP/IP connection after the file contents are sent.

This decreases the packet overhead for future connections.

The Server Response Header

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
Cache-Control: no-cache
Content-Type: text/html
Content-Encoding: gzip
Content-Length: 9062
```
Followed by data from file, in this case encoded using gzip

The above data is returned by the server, to the client:

The HTTP/1.1 200 OK line tells the client/browser that HTTP1.1 is supported, and the 200 tells the client that the file was found

The Server line informs the client of the Web Server type and version

The Cache-Control line tells the client to disable cache

The Content-Type line tells the client the type of data that will follow

The Content-Encoding line tells the client that the following data is encrypted using gzip

The Content-Length line tells the client how many bytes are to follow

HTTP 1.1

HTTP 1.1 is defined by RFC2616

Additions to HTTP 1.1:

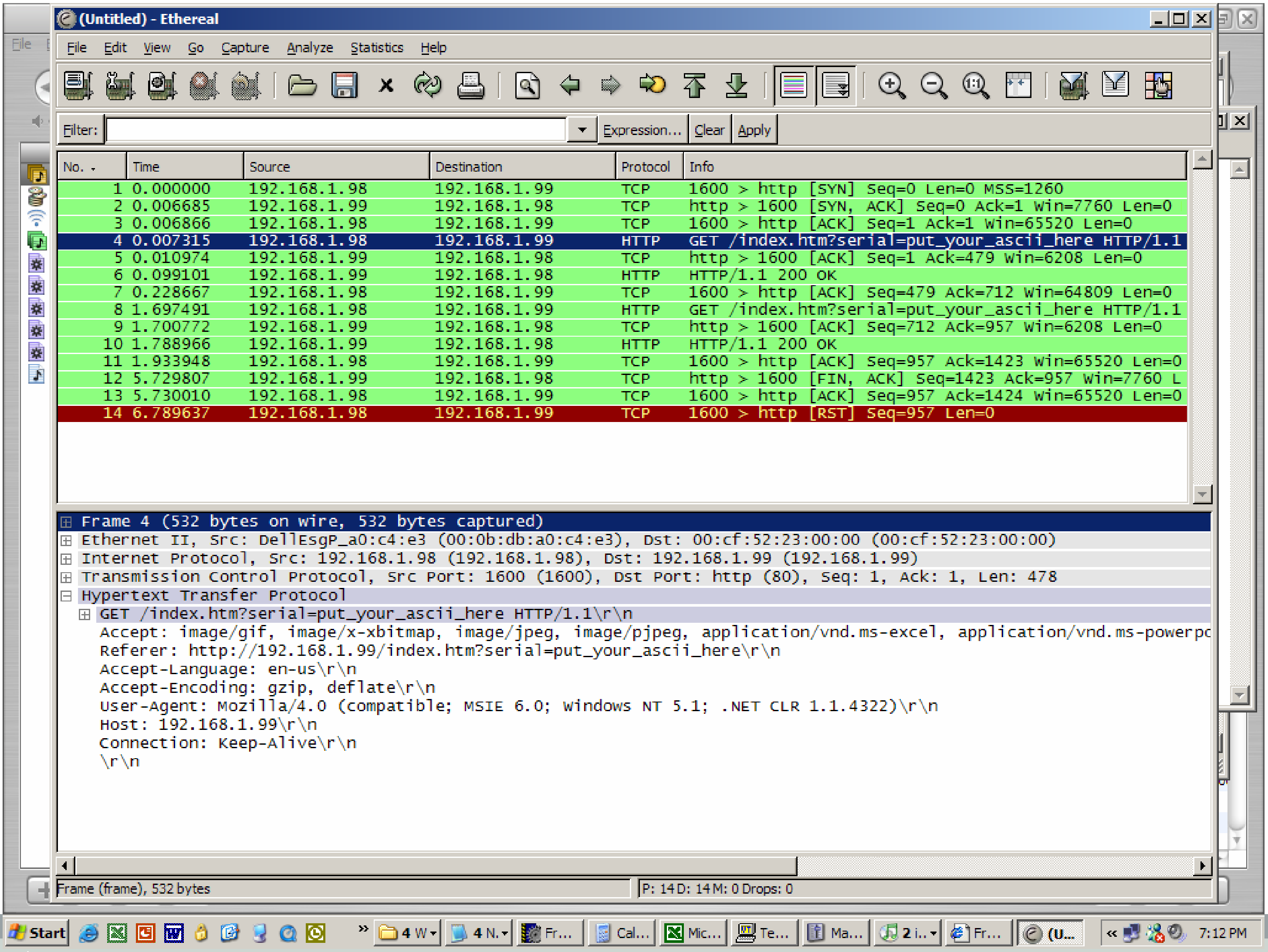
Faster response, by allowing multiple transactions to take place over a single *persistent connection*.

Faster response and great bandwidth savings, by adding cache support.

Faster response for dynamically-generated pages, by supporting *chunked encoding*, which allows a response to be sent before its total length is known.

Efficient use of IP addresses, by allowing multiple domains to be served from a single IP address.

Ethereal HTTP demo



(Untitled) - Ethereal

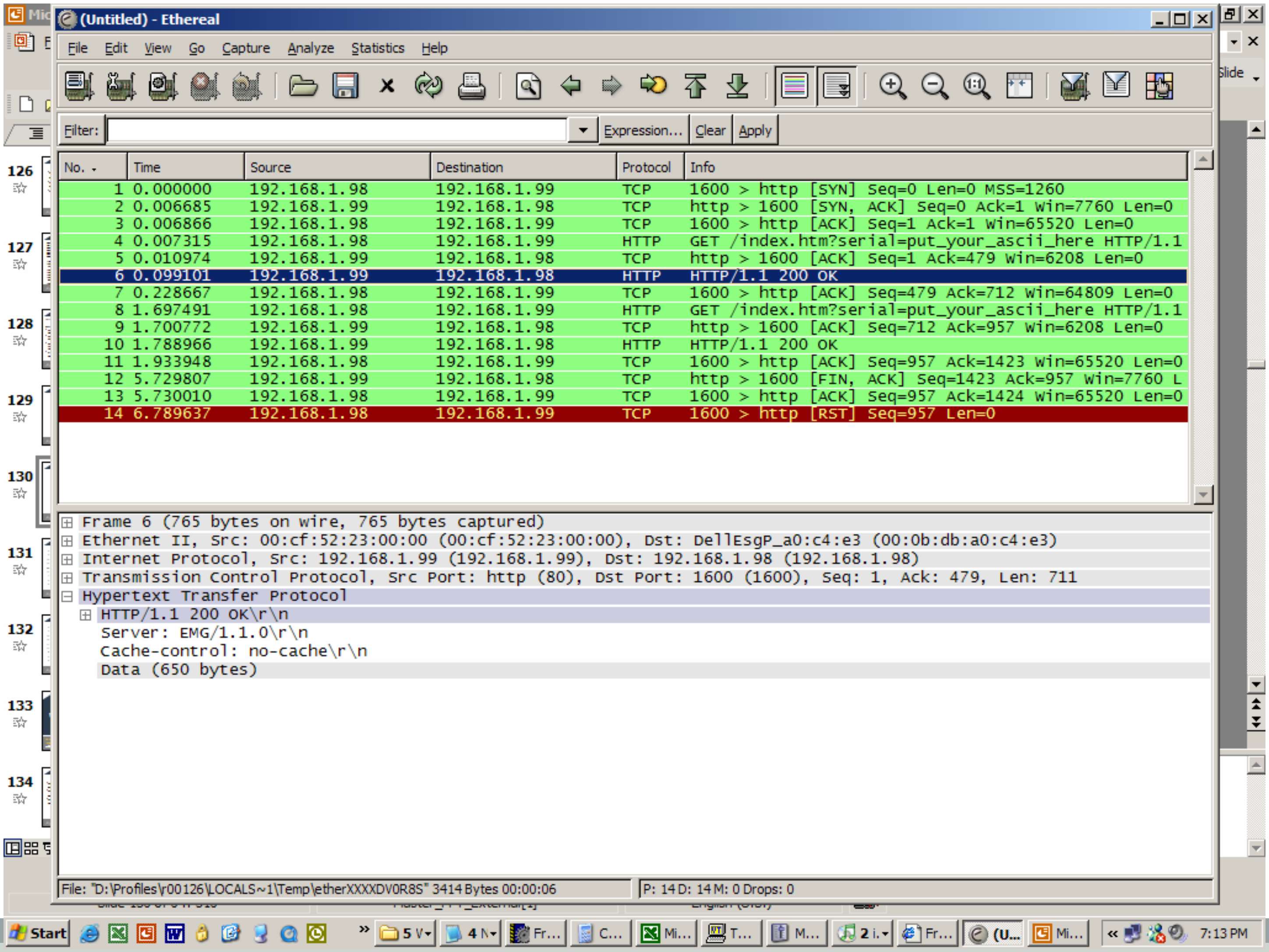
Filter:

```
No. Time       Source        Destination   Protocol Info
  1 0.000000   192.168.1.98  192.168.1.99  TCP      1600 > http [SYN] Seq=0 Len=0 MSS=1260
  2 0.006685   192.168.1.99  192.168.1.98  TCP      http > 1600 [SYN, ACK] Seq=0 Ack=1 Win=7760 Len=0
  3 0.006866   192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=1 Ack=1 Win=65520 Len=0
  4 0.007315   192.168.1.98  192.168.1.99  HTTP     GET /index.htm?serial=put_your_ascii_here HTTP/1.1
  5 0.010974   192.168.1.99  192.168.1.98  TCP      http > 1600 [ACK] Seq=1 Ack=479 Win=6208 Len=0
  6 0.099101   192.168.1.99  192.168.1.98  HTTP     HTTP/1.1 200 OK
  7 0.228667   192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=479 Ack=712 Win=64809 Len=0
  8 1.697491   192.168.1.98  192.168.1.99  HTTP     GET /index.htm?serial=put_your_ascii_here HTTP/1.1
  9 1.700772   192.168.1.99  192.168.1.98  TCP      http > 1600 [ACK] Seq=712 Ack=957 Win=6208 Len=0
 10 1.788966   192.168.1.99  192.168.1.98  HTTP     HTTP/1.1 200 OK
 11 1.933948   192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=957 Ack=1423 Win=65520 Len=0
 12 5.729807   192.168.1.99  192.168.1.98  TCP      http > 1600 [FIN, ACK] Seq=1423 Ack=957 Win=7760 L
 13 5.730010   192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=957 Ack=1424 Win=65520 Len=0
 14 6.789637   192.168.1.98  192.168.1.99  TCP      1600 > http [RST] Seq=957 Len=0
```

```
⊞ Frame 4 (532 bytes on wire, 532 bytes captured)
⊞ Ethernet II, Src: DellEsgP_a0:c4:e3 (00:0b:db:a0:c4:e3), Dst: 00:cf:52:23:00:00 (00:cf:52:23:00:00)
⊞ Internet Protocol, Src: 192.168.1.98 (192.168.1.98), Dst: 192.168.1.99 (192.168.1.99)
⊞ Transmission Control Protocol, Src Port: 1600 (1600), Dst Port: http (80), Seq: 1, Ack: 1, Len: 478
⊟ Hypertext Transfer Protocol
  ⊞ GET /index.htm?serial=put_your_ascii_here HTTP/1.1\r\n
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpc
    Referer: http://192.168.1.99/index.htm?serial=put_your_ascii_here\r\n
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)\r\n
    Host: 192.168.1.99\r\n
    Connection: Keep-Alive\r\n
    \r\n
```

Frame (frame), 532 bytes | P: 14 D: 14 M: 0 Drops: 0

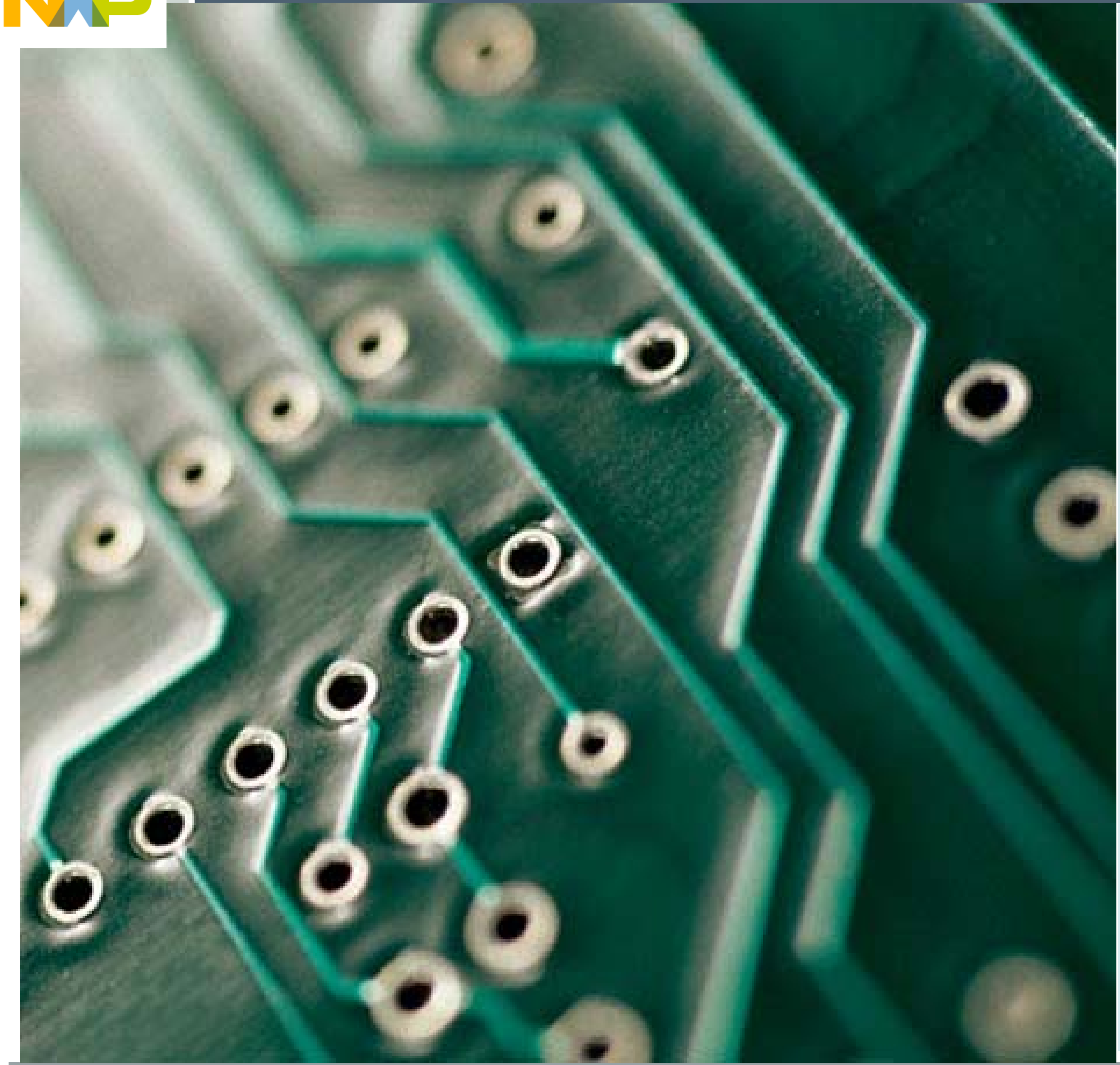
Ethereal HTTP demo


(Untitled) - Ethereal

```
No. - Time      Source        Destination   Protocol Info
    1 0.000000  192.168.1.98  192.168.1.99  TCP      1600 > http [SYN] Seq=0 Len=0 MSS=1260
    2 0.006685  192.168.1.99  192.168.1.98  TCP      http > 1600 [SYN, ACK] Seq=0 Ack=1 Win=7760 Len=0
    3 0.006866  192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=1 Ack=1 Win=65520 Len=0
    4 0.007315  192.168.1.98  192.168.1.99  HTTP     GET /index.htm?serial=put_your_ascii_here HTTP/1.1
    5 0.010974  192.168.1.99  192.168.1.98  TCP      http > 1600 [ACK] Seq=1 Ack=479 Win=6208 Len=0
    6 0.099101  192.168.1.99  192.168.1.98  HTTP     HTTP/1.1 200 OK
    7 0.228667  192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=479 Ack=712 Win=64809 Len=0
    8 1.697491  192.168.1.98  192.168.1.99  HTTP     GET /index.htm?serial=put_your_ascii_here HTTP/1.1
    9 1.700772  192.168.1.99  192.168.1.98  TCP      http > 1600 [ACK] Seq=712 Ack=957 Win=6208 Len=0
   10 1.788966  192.168.1.99  192.168.1.98  HTTP     HTTP/1.1 200 OK
   11 1.933948  192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=957 Ack=1423 Win=65520 Len=0
   12 5.729807  192.168.1.99  192.168.1.98  TCP      http > 1600 [FIN, ACK] Seq=1423 Ack=957 Win=7760 L
   13 5.730010  192.168.1.98  192.168.1.99  TCP      1600 > http [ACK] Seq=957 Ack=1424 Win=65520 Len=0
   14 6.789637  192.168.1.98  192.168.1.99  TCP      1600 > http [RST] Seq=957 Len=0
```

```
⊞ Frame 6 (765 bytes on wire, 765 bytes captured)
⊞ Ethernet II, Src: 00:cf:52:23:00:00 (00:cf:52:23:00:00), Dst: DellEsgP_a0:c4:e3 (00:0b:db:a0:c4:e3)
⊞ Internet Protocol, Src: 192.168.1.99 (192.168.1.99), Dst: 192.168.1.98 (192.168.1.98)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 1600 (1600), Seq: 1, Ack: 479, Len: 711
⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.1 200 OK\r\n
    Server: EMG/1.1.0\r\n
    Cache-control: no-cache\r\n
    Data (650 bytes)
```

File: "D:\Profiles\r00126\LOCALS~1\Temp\etherXXXXDV0R8S" 3414 Bytes 00:00:06 | P: 14 D: 14 M: 0 Drops: 0



Customer Web Server Labs

Loading the Default Static Web Page

The purpose of this lab is to use CodeWarrior® to build and load the stack and default web page.

The static file system utility will be used to change the default static web page.

We will also learn how to configure the static IP address in both the demo board and the PC.

Using CodeWarrior® to Build the Default Web Page

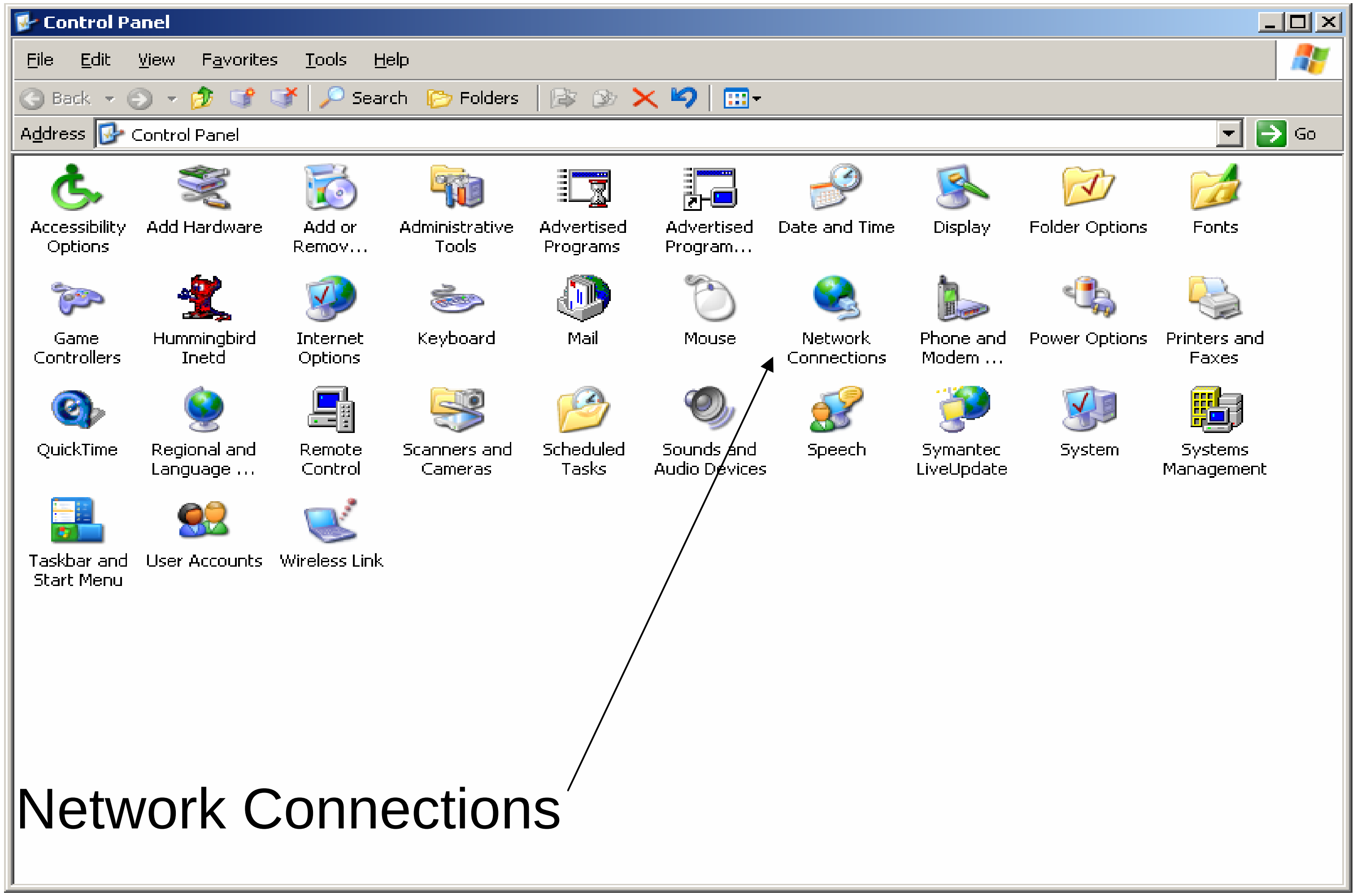
Follow the instructions from the CodeWarrior lab to configure CodeWarrior and the flash programmer for the MCF5223x.

Load the MCP file

Set up PC Network Connection

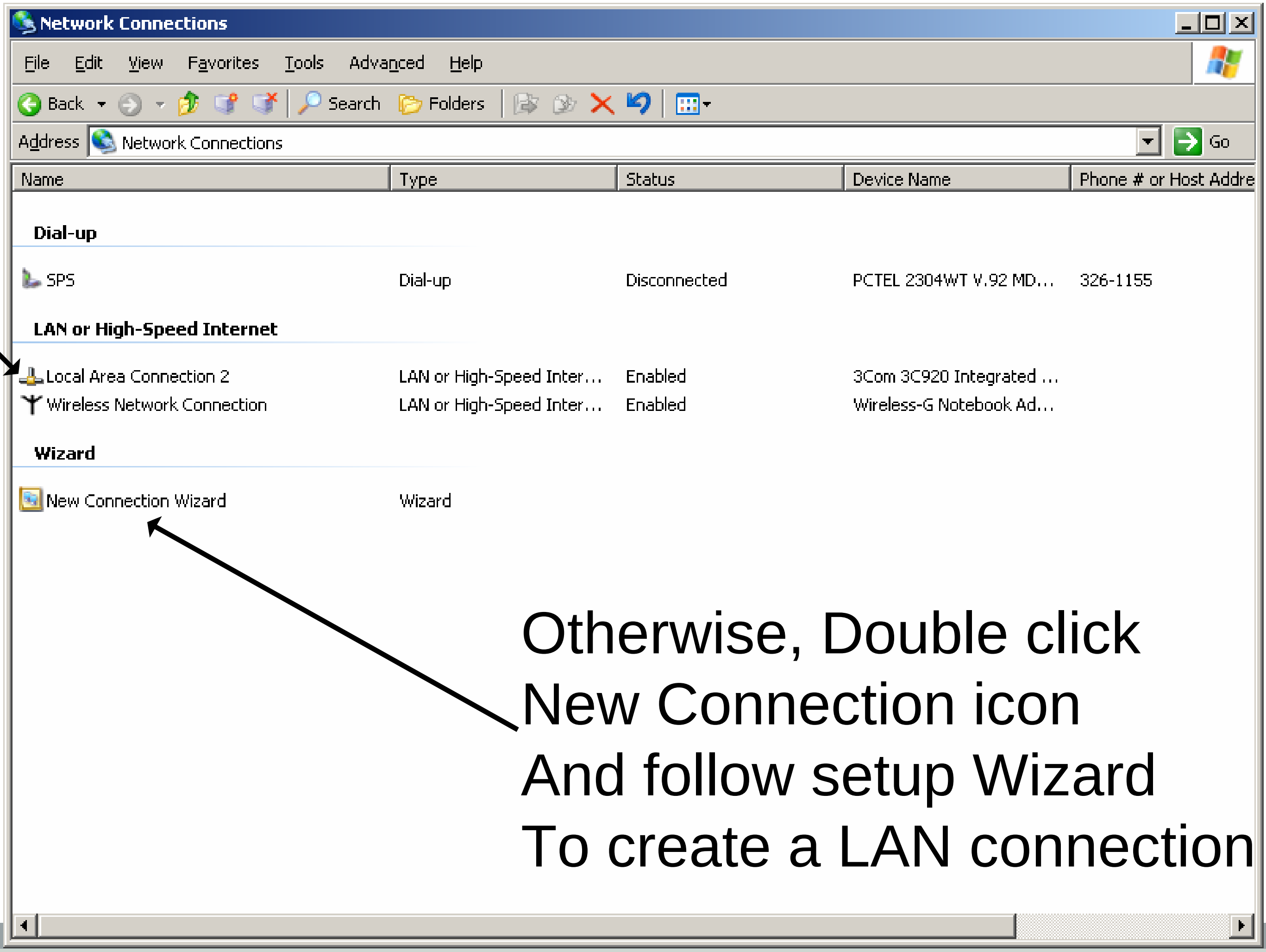
Follow one of the following two methods:

- From Control Panel install new connect.
- Use existing connection.



Double click Network Connections

If available, Double click icon.



Name	Type	Status	Device Name	Phone # or Host Addre
Dial-up				
SPS	Dial-up	Disconnected	PCTEL 2304WT V.92 MD...	326-1155
LAN or High-Speed Internet				
Local Area Connection 2	LAN or High-Speed Inter...	Enabled	3Com 3C920 Integrated ...	
Wireless Network Connection	LAN or High-Speed Inter...	Enabled	Wireless-G Notebook Ad...	
Wizard				
New Connection Wizard	Wizard			

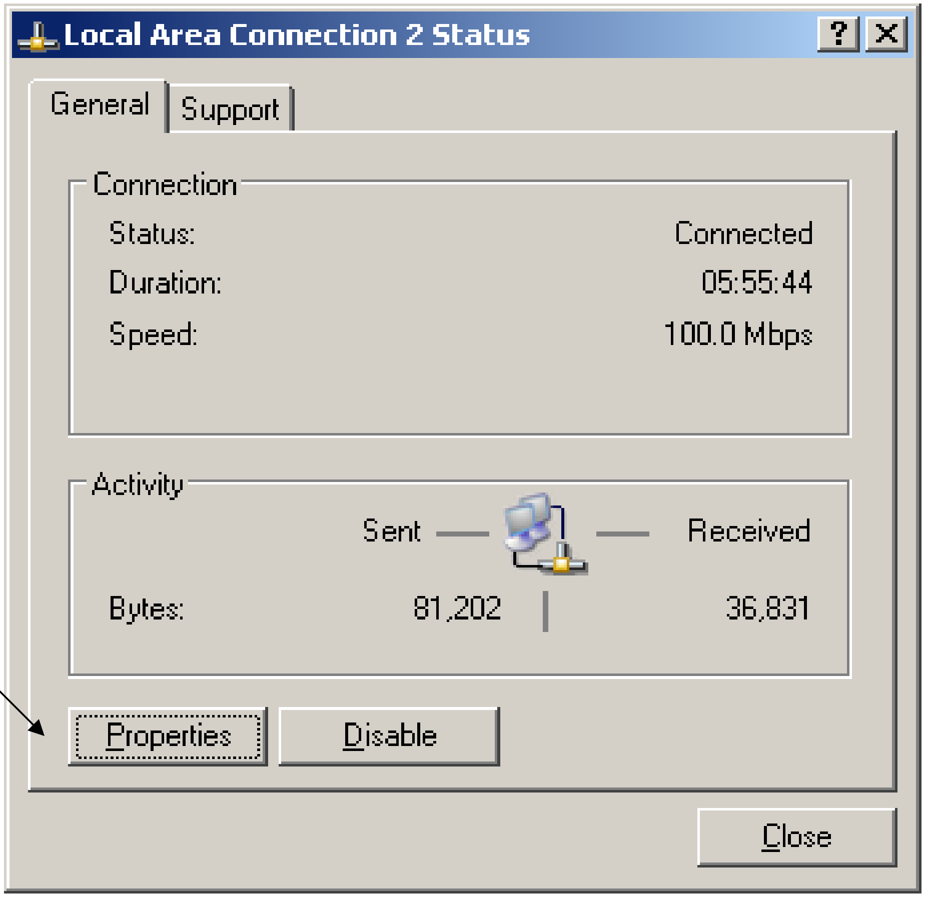
Network Connections
File Edit View Favorites Tools Advanced Help
Back Search Folders
Address Network Connections
Go

Otherwise, Double click New Connection icon And follow setup Wizard To create a LAN connection

Now that a LAN connection is available

Let's set it up for our needs

Click on Properties Tab


Local Area Connection 2 Status

Connection

Status: Connected

Duration: 05:55:44

Speed: 100.0 Mbps

Activity

Sent — Received

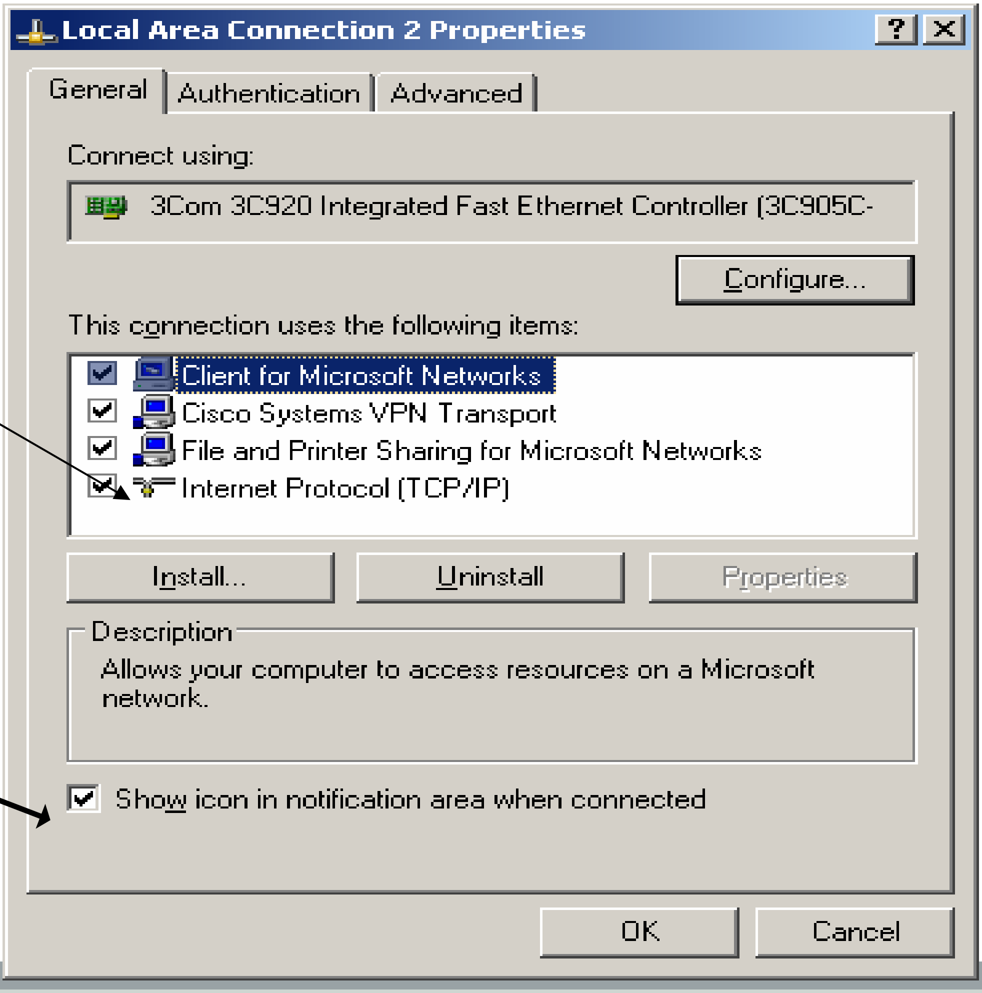
Bytes: 81,202 | 36,831

Set up PC Network Connection

The following properties dialog will open

Double Click on the Internet Protocol (TCP/IP) Icon

Checking this will aid config changes later



Local Area Connection 2 Properties

General | Authentication | Advanced

Connect using:

3Com 3C920 Integrated Fast Ethernet Controller (3C905C-

Configure...

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [✓] Cisco Systems VPN Transport
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol (TCP/IP)

Install... | Uninstall | Properties

Description

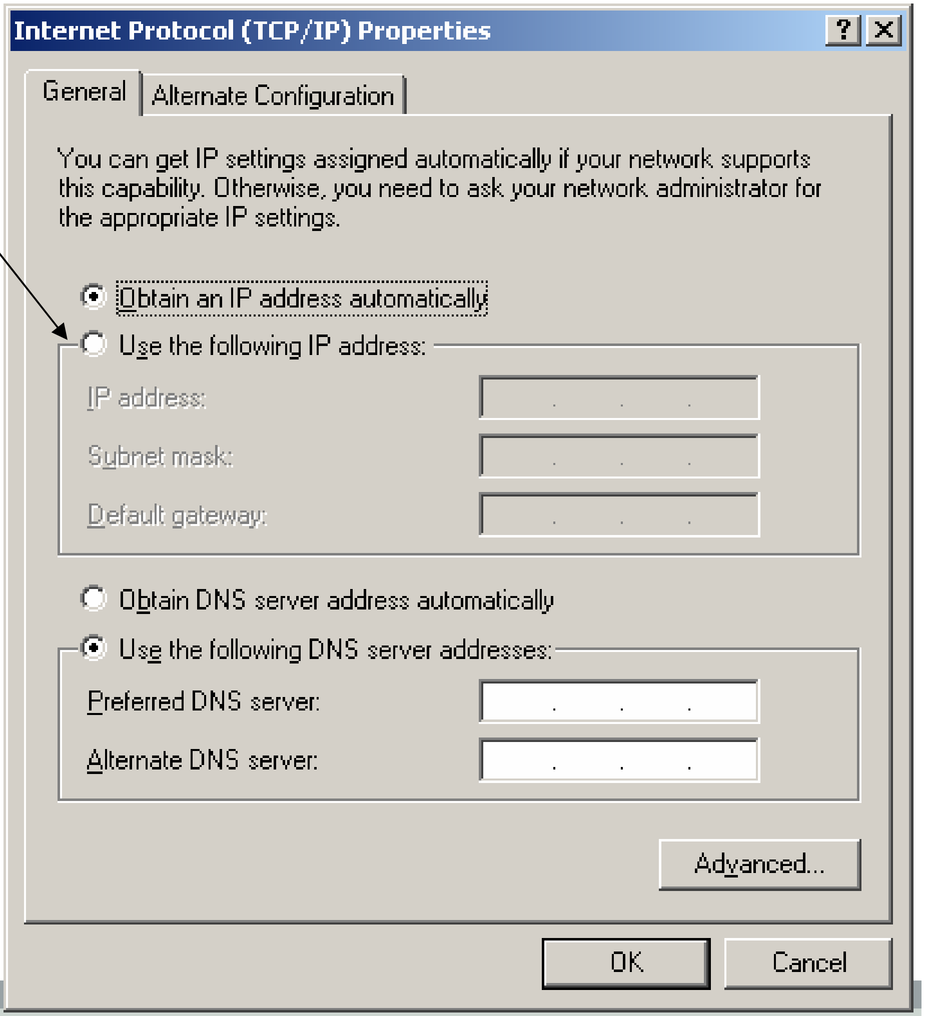
Allows your computer to access resources on a Microsoft network.

[✓] Show icon in notification area when connected

OK | Cancel

Set up PC Network Connection

Select
Use the following IP address



Internet Protocol (TCP/IP) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]

Use the following IP address:

IP address: . . .

Subnet mask: . . .

Default gateway: . . .

Obtain DNS server address automatically

[✓ Use the following DNS server addresses:]

Preferred DNS server: . . .

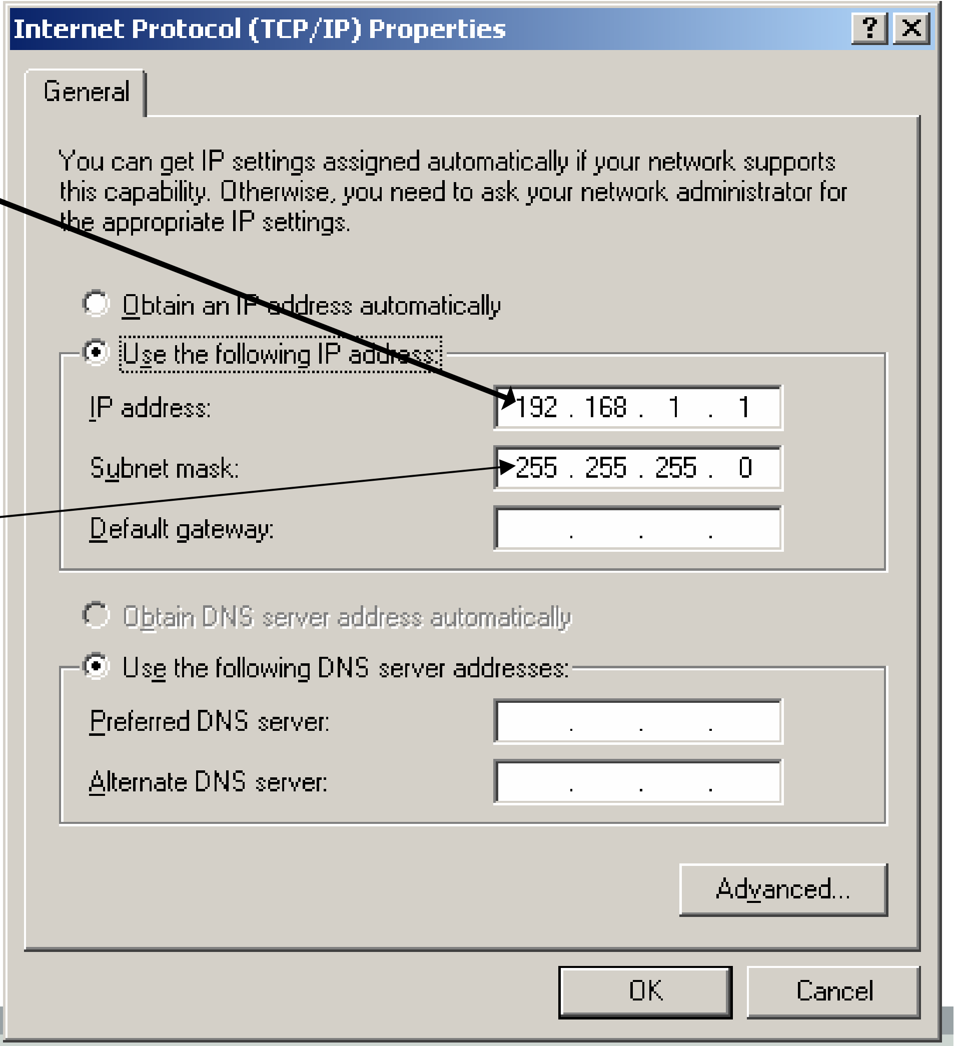
Alternate DNS server: . . .

Set up PC Network Connection

Enter 192.168.1.1 for the IP address

Click in the Subnet Mask Field and it will auto-fill with 255.255.255.0

Click OK on all LAN setup dialog boxes and close them



Internet Protocol (TCP/IP) Properties

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

- ○ Obtain an IP address automatically
- ● Use the following IP address:

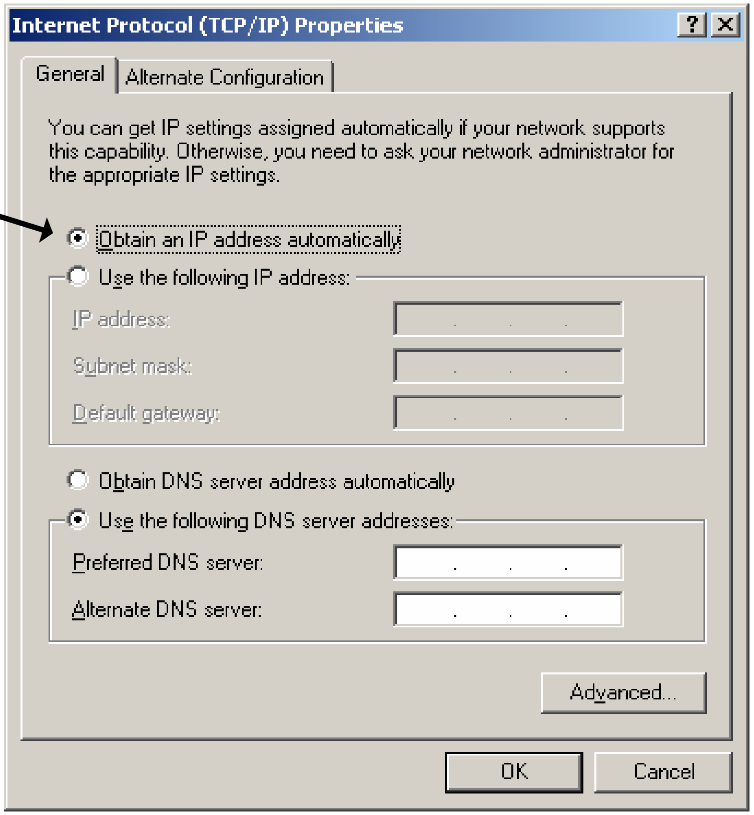
IP address:	192 . 168 . 1 . 1
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

- ○ Obtain DNS server address automatically
- ● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

Set up PC Network Connection

To return your LAN setting for normal Operation reopen the Internet Properties Dialog box and select Auto IP address


Internet Protocol (TCP/IP) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

● Obtain an IP address automatically
○ Use the following IP address:

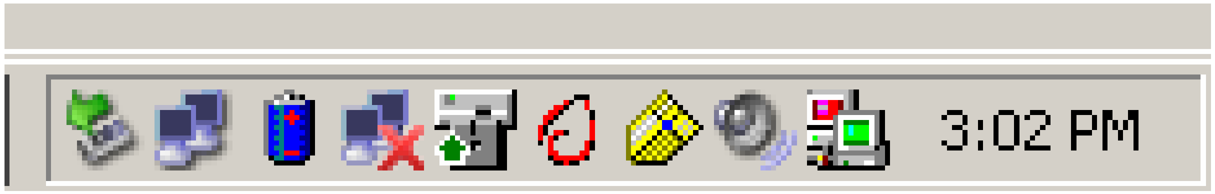
IP address:	. . .
Subnet mask:	. . .
Default gateway:	. . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

The Taskbar

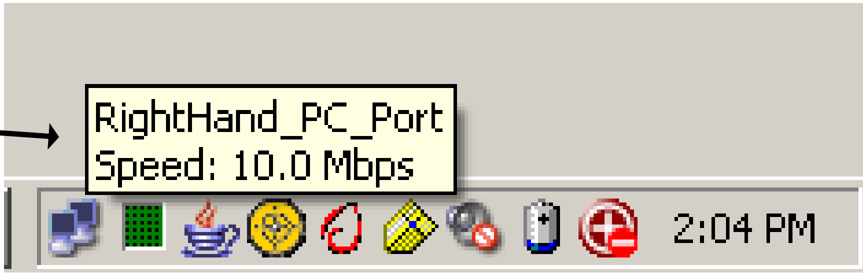
If there was a connection previously, an icon may be on the taskbar



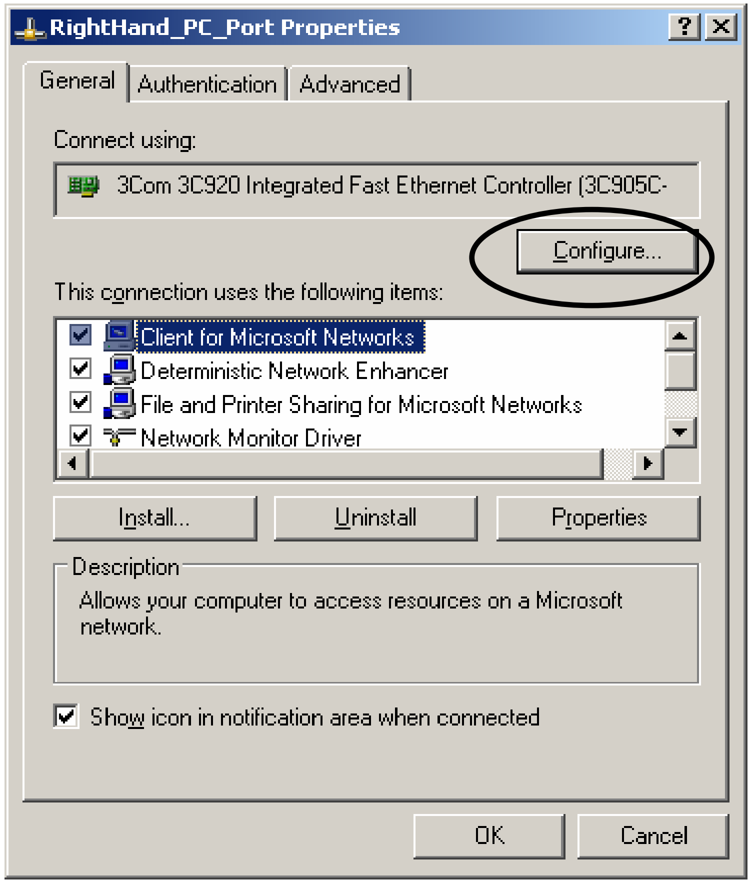
Configured LAN connections

If a connection bubble like shown at left is not shown (100bT is OK too)

And you have a 1gbit card reconfigure your card as shown in the next slides



Setting Speed to 100Mb, Half for 1Gbit Cards



RightHand_PC_Port Properties

General | Authentication | Advanced

Connect using:

3Com 3C920 Integrated Fast Ethernet Controller (3C905C-

Configure...

This connection uses the following items:

- ✓ Client for Microsoft Networks
- ✓ Deterministic Network Enhancer
- ✓ File and Printer Sharing for Microsoft Networks
- ✓ Network Monitor Driver

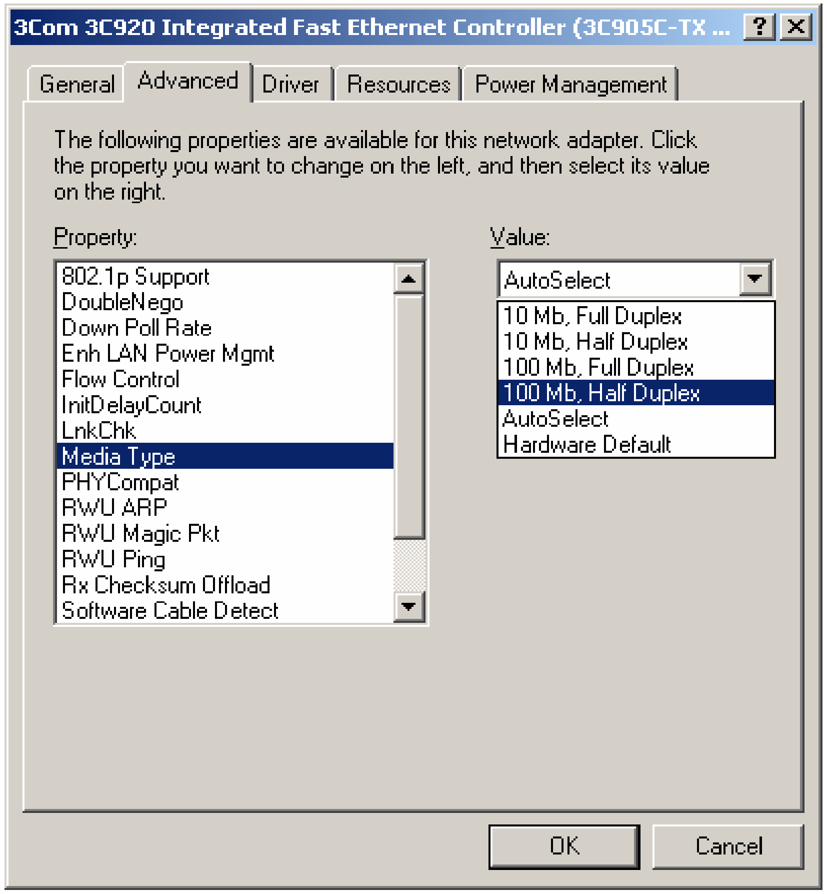
Install... | Uninstall | Properties

Description

Allows your computer to access resources on a Microsoft network.

✓ Show icon in notification area when connected

OK | Cancel



3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX ...

General | Advanced | Driver | Resources | Power Management

The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:
- 802.1p Support
- DoubleNego
- Down Poll Rate
- Enh LAN Power Mgmt
- Flow Control
- InitDelayCount
- LnkChk
- Media Type
- PHYCompat
- RWU ARP
- RWU Magic Pkt
- RWU Ping
- Rx Checksum Offload
- Software Cable Detect

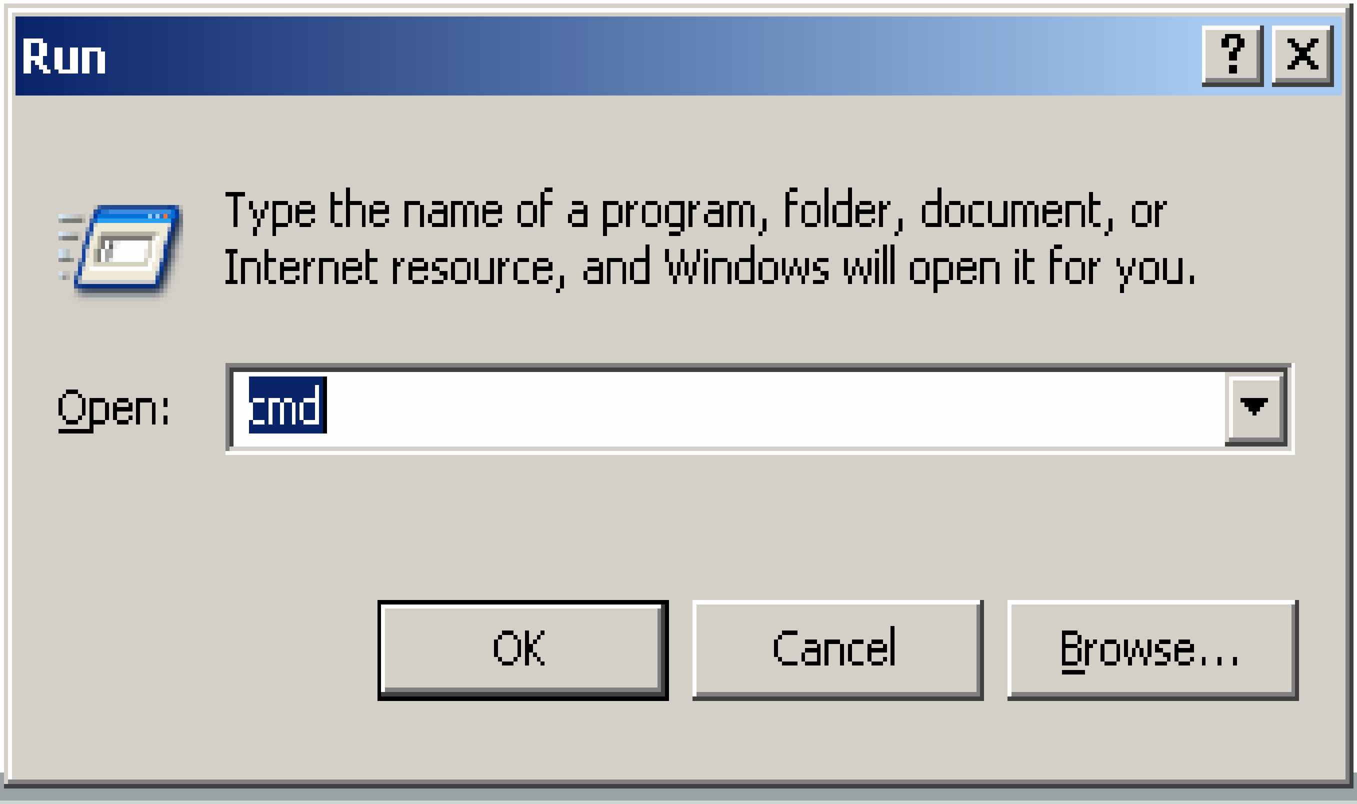
Value: AutoSelect
- 10 Mb, Full Duplex
- 10 Mb, Half Duplex
- 100 Mb, Full Duplex
- 100 Mb, Half Duplex
- AutoSelect
- Hardware Default

OK | Cancel

*****Only needed if communications issues with 1 Gbit card

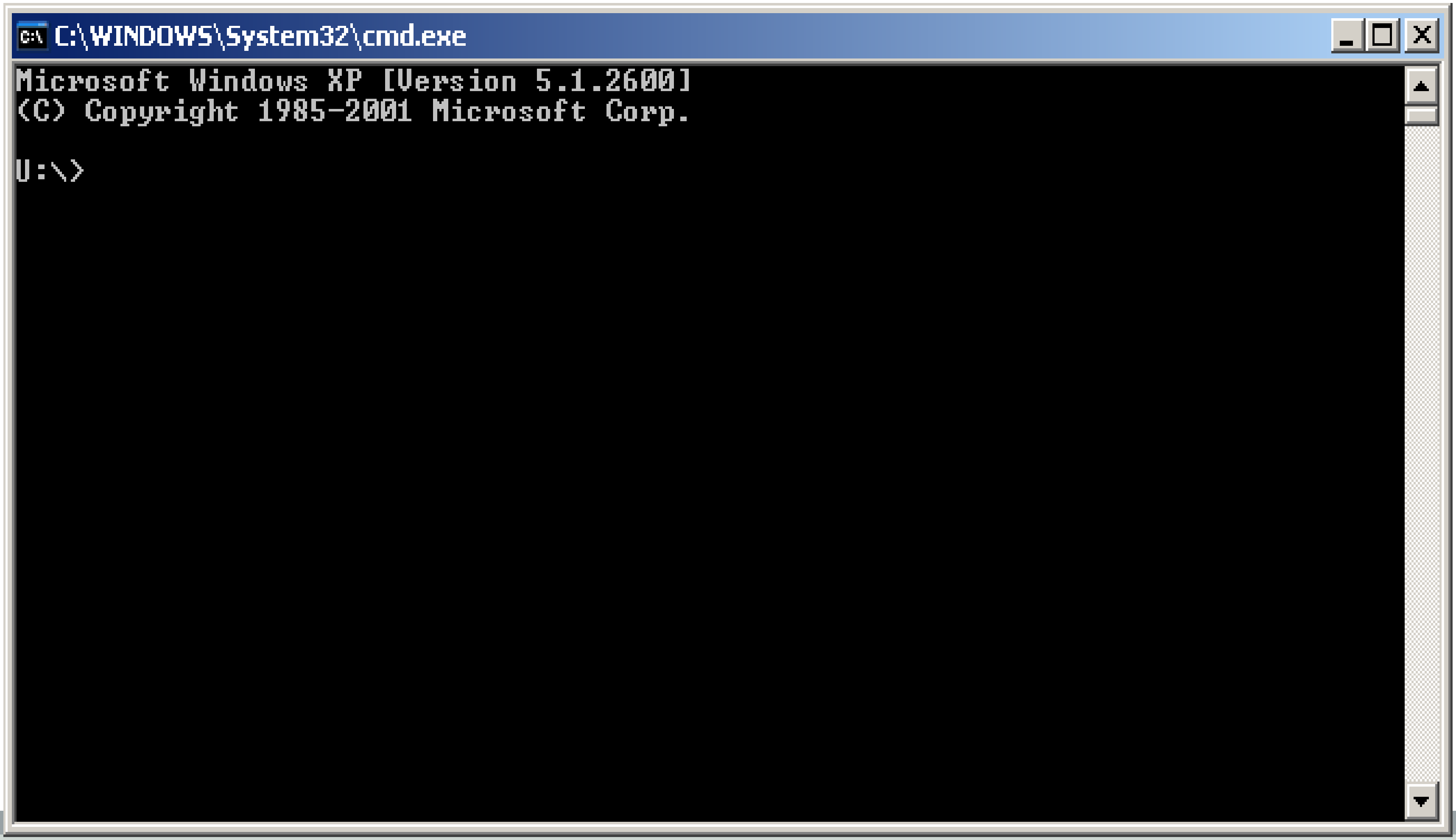
RUN

From the Start menu select RUN
Enter “CMD”, click OK


Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

DOS Window

A DOS window should open.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

U:\>
```

Enter -> ping 192.168.1.99

At the DOS prompt type   ping 192.168.1.99   then hit enter

If Ping Does not work

Go to your hyperterminal window, hit enter a few times.
Verify a INET> prompt appears.

Verify that you have a cross connect cable.

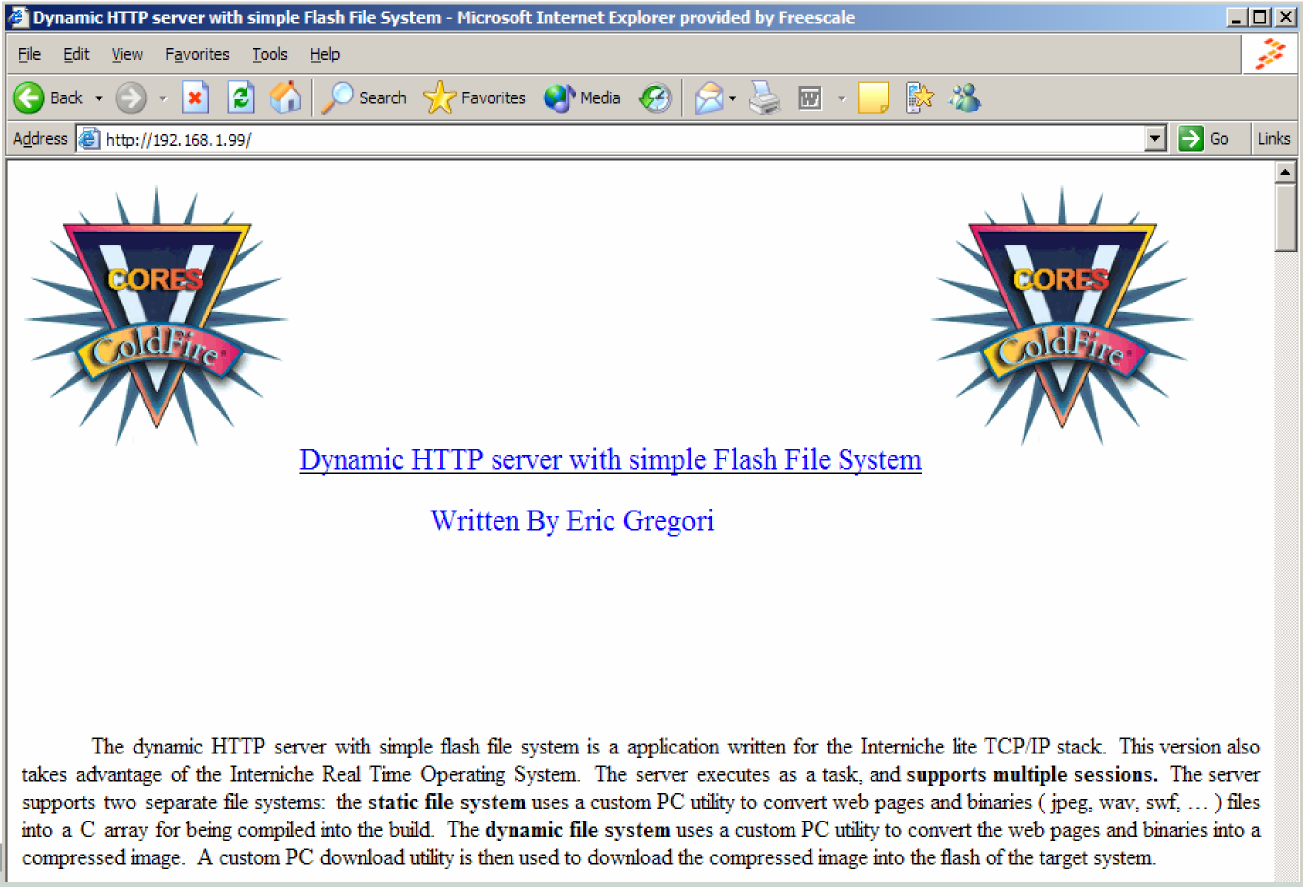
Verify that you have disabled VPN (on your personal machine)

Type   iface soft   at the INET> prompt.

Try Ping again after 2 seconds.

The Default Web Page

Open Internet Explorer, and type 192.168.1.99 (the IP address of the demo board) into the address bar. This is the default compile time web page you just loaded with the TCP/IP stack and Web Server.


Dynamic HTTP server with simple Flash File System - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://192.168.1.99/
Go
Links
CORES
ColdFire

Dynamic HTTP server with simple Flash File System

Written By Eric Gregori

The dynamic HTTP server with simple flash file system is a application written for the Interniche lite TCP/IP stack. This version also takes advantage of the Interniche Real Time Operating System. The server executes as a task, and **supports multiple sessions.** The server supports two separate file systems: the **static file system** uses a custom PC utility to convert web pages and binaries (jpeg, wav, swf, …) files into a C array for being compiled into the build. The **dynamic file system** uses a custom PC utility to convert the web pages and binaries into a compressed image. A custom PC download utility is then used to download the compressed image into the flash of the target system.

The Static/Compile Time Flash File System

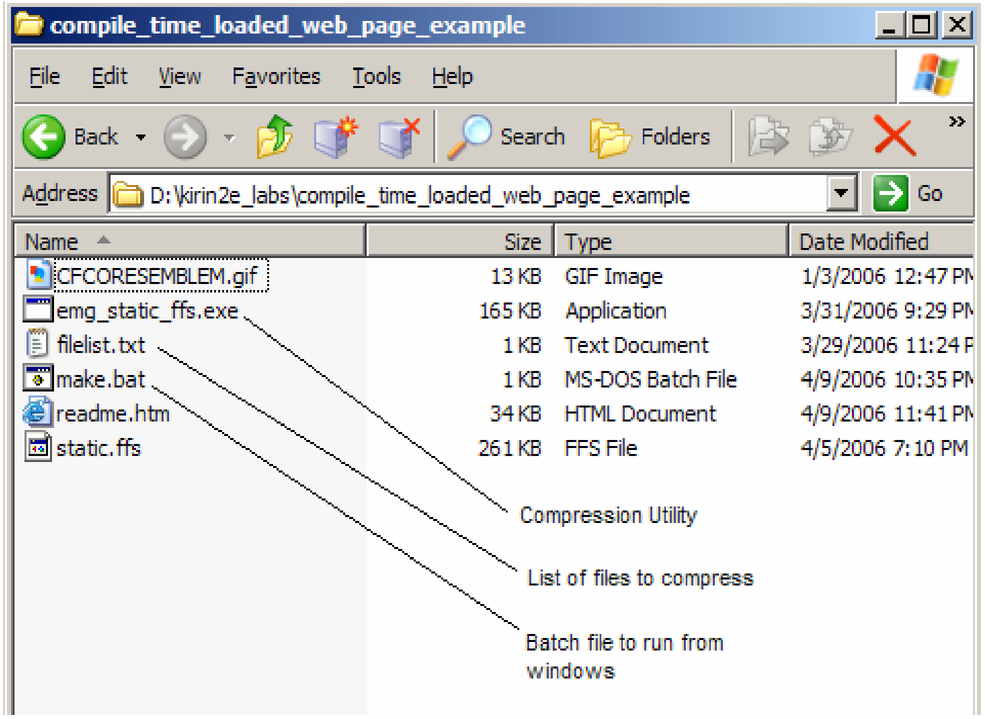
The Static/Compile Time Flash File System allows the user to embed web pages consisting of one or multiple files into a target build.

The system has two parts: The firmware running in the ***ColdFire***® processor as part of the Web Server, and the compression utility which is executed on the PC.

The Compression utility takes a list of files, and compresses them into a single ‘C’ file. The ‘C’ file is then compiled and linked into the final target build with the TCP/IP stack and the Web Server.

The Static/Compile Time Flash File System Compression Utility

The compression utility: emg_static_ffs.exe is a DOS command utility that can be executed from windows using a BATCH file.


compile_time_loaded_web_page_example

Address D:\kirin2e_labs\compile_time_loaded_web_page_example

Name	Size	Type	Date Modified
CFCORESEMBLEM.gif	13 KB	GIF Image	1/3/2006 12:47 PM
emg_static_ffs.exe	165 KB	Application	3/31/2006 9:29 PM
filelist.txt	1 KB	Text Document	3/29/2006 11:24 P
make.bat	1 KB	MS-DOS Batch File	4/9/2006 10:35 PM
readme.htm	34 KB	HTML Document	4/9/2006 11:41 PM
static.ffs	261 KB	FFS File	4/5/2006 7:10 PM

Compression Utility Command Line Arguments

Emg_static_ffs filelist.txt output_file.c

Where:

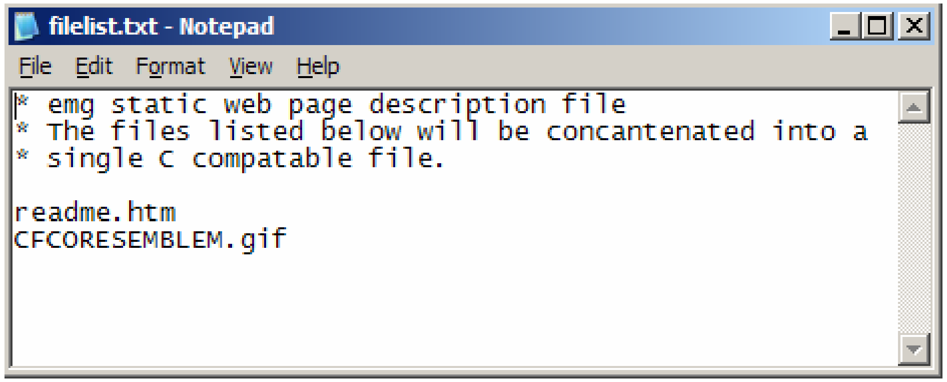
Filelist.txt is a text file containing the list of files to compress. Each file should be on its own line, and the first file is the default.

Comments can be added using a '*' as the first character in a line.

Output_file.c is the file generated containing all the files in the filelist compressed together, along with data structures used to reference the files from the Web Server.

Sample filelist.txt

```
* emg static web page description file
* The files listed below will be concantenated into a
* single C compatable file.

readme.htm
CFCORESEMBLEM.gif
```



```
* emg static web page description file
* The files listed below will be concantenated into a
* single C compatable file.

readme.htm
CFCORESEMBLEM.gif
```

The last line must be a blank line with just a CRLF (just hit enter in the last blank line).

The output_file.c

```
//*************************************************************************
//* Static Flash File System Generator                          *
//* Written by Eric Gregori - Chicago FAE                        *
//*                                              *
//*************************************************************************


    const unsigned char readme_htm[] = {
    0x48,0x54,0x54,0x50,0x2F,0x31,0x2E,0x31,0x20,0x32,

    Data removed for space in presentation

    0x0D,0x0A,0x00 };

    const unsigned char CFCORESEMBLEM_gif[] = {
    0x48,0x54,0x54,0x50,0x2F,0x31,0x2E,0x31,0x20,0x32,

    Data removed for space in presentation

    0xA8,0xC7,0x3D,0xF2,0xB1,0x8F,0x7E,0xFC,0x23,0x20,
    0x6F,0x17,0x10,0x00,0x3B,0x00 };

    const char *emg_static_ffs_filenames[] =   {
                                    "readme.htm",
                                    "CFCORESEMBLEM.gif"
                                  };

    const unsigned char *emg_static_ffs_ptrs[] =   {
                                    readme_htm,
                                    CFCORESEMBLEM_gif
                                  };

    const unsigned short emg_static_ffs_len[] =     {
                                    34506,
                                    12919
                                  };

    const unsigned char emg_static_ffs_type[] =     {
                      0x68746d6c,
                         0x67696666
                                  };

    const unsigned char emg_static_ffs_nof = 2;
```

The output file contains the contents of each file stored as a ‘C’ array. The files inserted are from the filelist.txt file (see previous slides).

Array containing list of filenames

Array containing list of pointers to files.

Array containing file sizes

Array containing file type

Number of files

Other Uses for the Static/Compile Time Flash File System

User Data can also be stored in the static system. The data can be binary or text, but name the file *.txt. The utility actually treats all files as binary files.

The user can access the data from the firmware using examples in the firmware.

This feature can be usefull in the static/Compile Time System, but is considerably more usefull in the run time loadable system.

Static/Compile Time Web Page LAB 1

We are going to edit a HTML file.

Build a Compressed ‘C’ image.

Copy the Image to our project.

Re-build the project.

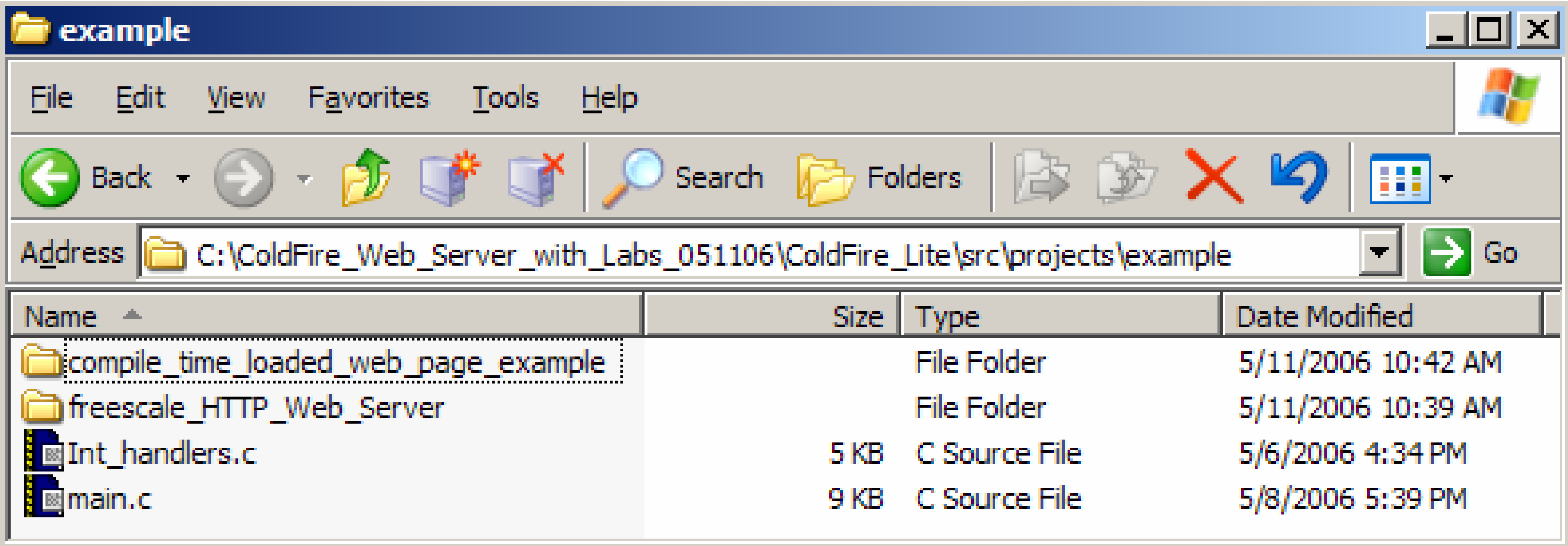
Load the new image in flash.

ColdFire_Lite Compile_Time_Loaded_Web_Page_Example

The Compile_Time_Loaded_Web_Page_Example

This is the directory for the static web page demo/lab.

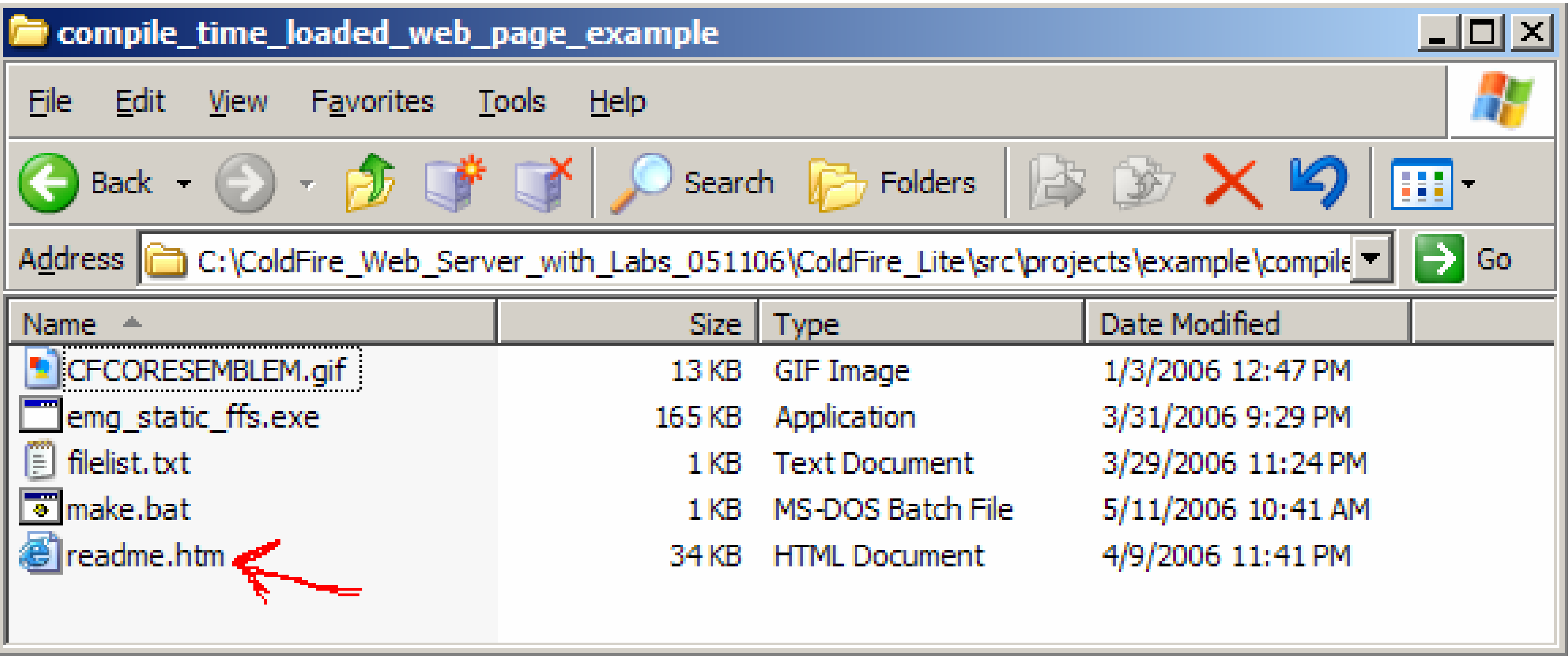
ColdFire_Lite\src\projects\example



Address C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\src\projects\example

Name	Size	Type	Date Modified
compile_time_loaded_web_page_example		File Folder	5/11/2006 10:42 AM
freescale_HTTP_Web_Server		File Folder	5/11/2006 10:39 AM
Int_handlers.c	5 KB	C Source File	5/6/2006 4:34 PM
main.c	9 KB	C Source File	5/8/2006 5:39 PM

Opening the HTML File


compile_time_loaded_web_page_example
File Edit View Favorites Tools Help
Back
Search
Folders
Address
Go

C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\src\projects\example\compile

Name	Size	Type	Date Modified
CFCORESEMBLEM.gif	13 KB	GIF Image	1/3/2006 12:47 PM
emg_static_ffs.exe	165 KB	Application	3/31/2006 9:29 PM
filelist.txt	1 KB	Text Document	3/29/2006 11:24 PM
make.bat	1 KB	MS-DOS Batch File	5/11/2006 10:41 AM
readme.htm	34 KB	HTML Document	4/9/2006 11:41 PM

HTML- An Overview

HTML or HyperText Markup Language is the language used to describe web pages.

HTML is a ascii text based language that defines how text and images are placed on a page.

HTML is a ascii text based language that uses “tags” to instruct a web browser how text and images are placed on a page.

HTML Tags

Tags start with a ‘<‘ and end with a ‘>’.

Most tags have a open and close form.

The open form <HTML>

The close form </HTML>

Tag form: <TAG ATTRIBUTE=value>

Tags/attributes are used to define placement, color, style, and fonts for text.

Tags are also used to define position and size for a image.

A Simple Web Page

```
<HTML>
<HEAD>
<TITLE>This text will appear at the top of the web browser, the navigation bar</TITLE>
</HEAD>
<BODY>
<CENTER>Hello World</CENTER>
</BODY>
</HTML>
```

The HTML element is used to tell the web browser that we are using HTML instead of JavaScript, or some other language.

The HEAD element contains meta-information. Meta-information is not part of the body of the document but defines the document in a general sense. The Title of the web page is a good example. It is not displayed in the body of the web page, but on the navigation bar of the web browser.

The BODY element defines the displayed portion of the web page.

Some Interesting HTML Tags

<CENTER>	Centers the object on the page.
<Hx>	Heading Size x, where x is from 1-6.
<P>	Start or paragraph.
<FONT COLOR=RED>	Sets font color to red.
<FONT SIZE=x>	Sets font size, where x is from 1-?.
<A HREF=“freescale.com”>	Makes text a URL pointing to freescale.com.
<IMG SRC=“filename.jpg”>	Puts the image filename.jpg into the web page.
<IMG SRC=filename.jpg” ALIGN=center>	Loads the image filename.jpg and centers it in the page.
<TABLE>	Creates a table with the help of <TR> table row and <TD> table data.

Creating Web Pages

Using notepad, you can start writing HTML immediately, and build your own Web Page.

Or, you can use an HTML generator.

- These programs allow you to design a web page, and generate the HTML for you.
- Just search for “HTML generator” on the web.
- There are dozens of them, some free.

Using Microsoft Word

Microsoft Word can also be used to generate a Web Page

By saving a document as *.htm in Microsoft Word, Word will create a web page.

- The web pages created by Word tend to be very large.
- Also, Word creates a subdirectory for images.
- Be sure to change the image reference paths to remove the directories.

The web page for this lab (readme.htm) was generated in Word.

Edit the HTML

To Edit the HTML open the readme.HTM file in Notepad

The first few lines of the readme.htm file

```
<html>
<head>
<title>Dynamic HTTP server with simple Flash File System</title>
</head>
```

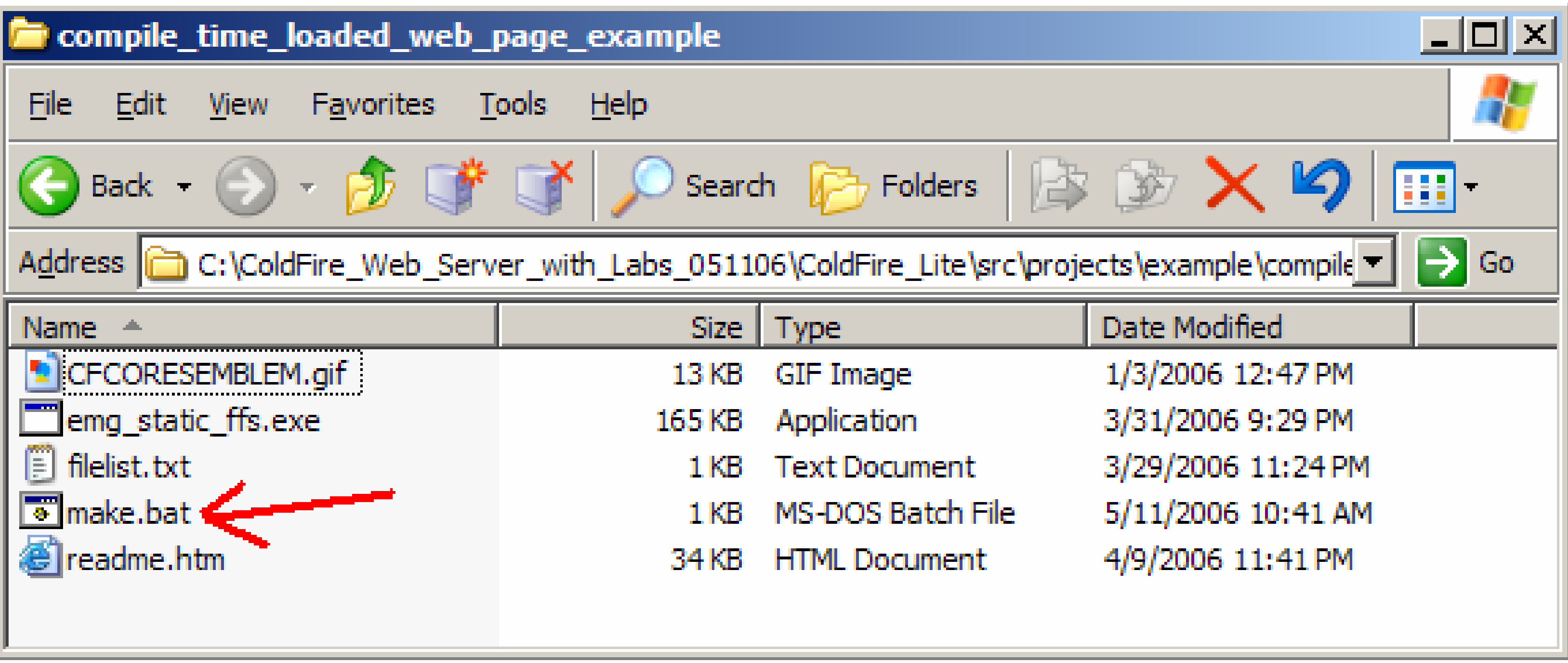
Modify the Dynamic HTTP server …. String with something else

```
<html>
<head>
<title>This is really cool</title>
</head>
```

Save the new file

Build a New Output File

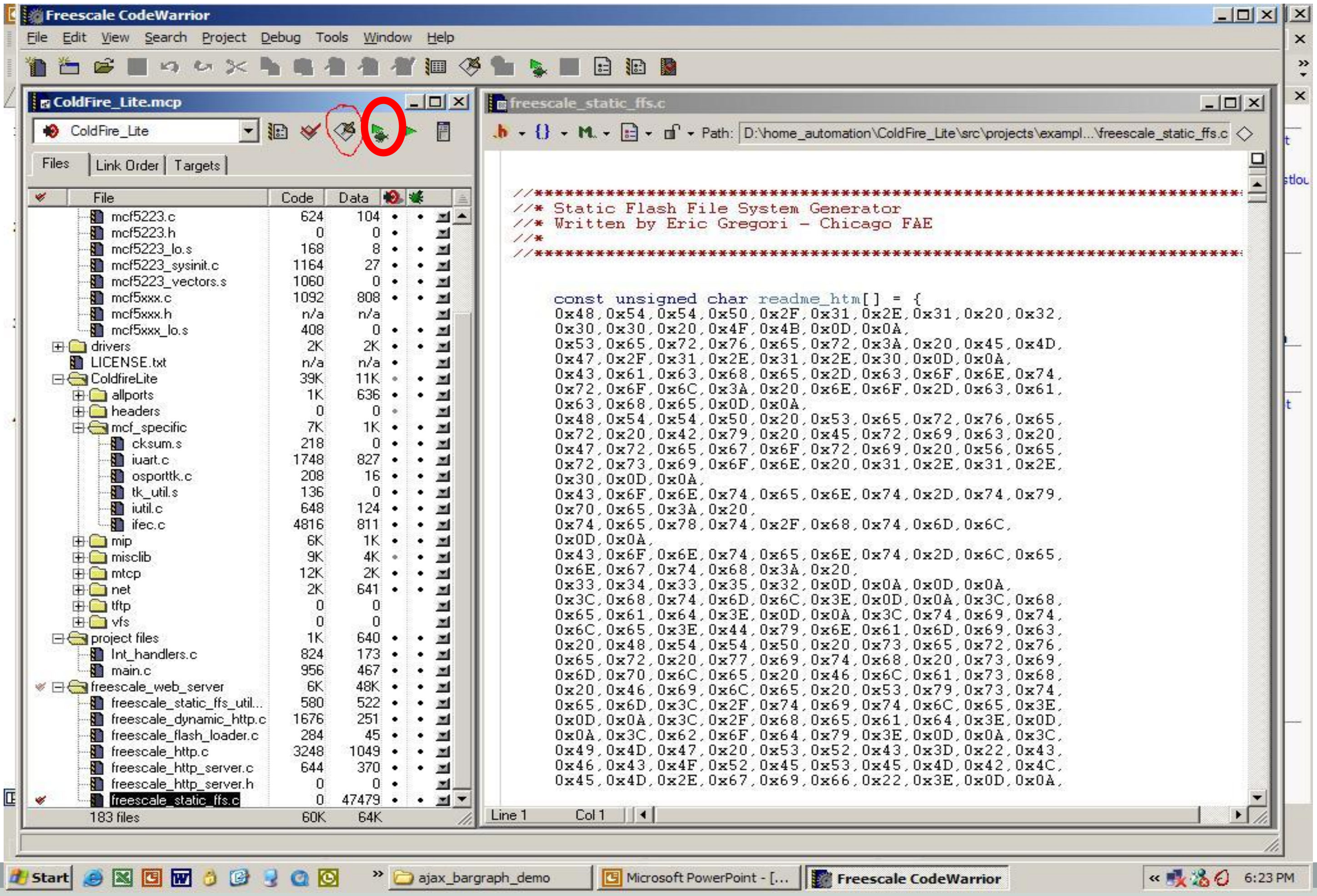
First Double click the batch file make.bat to build the image.



C:\ColdFire_Web_Server_with_Labs_051106\ColdFire_Lite\src\projects\example\compile

Name	Size	Type	Date Modified
CFCORESEMBLEM.gif	13 KB	GIF Image	1/3/2006 12:47 PM
emg_static_ffs.exe	165 KB	Application	3/31/2006 9:29 PM
filelist.txt	1 KB	Text Document	3/29/2006 11:24 PM
make.bat	1 KB	MS-DOS Batch File	5/11/2006 10:41 AM
readme.htm	34 KB	HTML Document	4/9/2006 11:41 PM

Build the project by clicking on the MAKE icon (circled in **RED**)

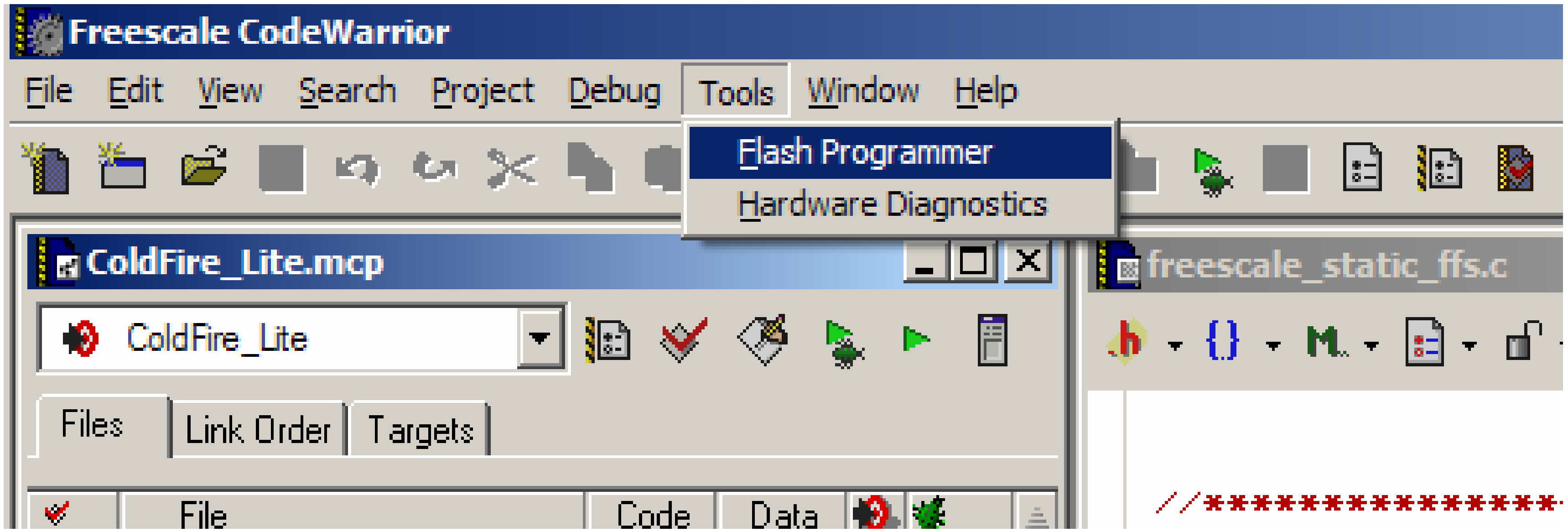

Freescale CodeWarrior
File Edit View Search Project Debug Tools Window Help
ColdFire_Lite.mcp
ColdFire_Lite
Files
Link Order
Targets
freescale_static_ffs.c
Path: D:\home_automation\ColdFire_Lite\src\projects\exampl...\freescale_static_ffs.c
Line 1
Col 1
Start
ajax_bargraph_demo
Microsoft PowerPoint - [...
Freescale CodeWarrior
6:23 PM

	File	Code	Data		
	mcf5223.c	624	104	●	●
	mcf5223.h	0	0	●	
	mcf5223_lo.s	168	8	●	●
	mcf5223_sysinit.c	1164	27	●	●
	mcf5223_vectors.s	1060	0	●	●
	mcf5xxx.c	1092	808	●	●
	mcf5xxx.h	n/a	n/a		
	mcf5xxx_lo.s	408	0	●	●
	drivers	2K	2K	●	●
	LICENSE.txt	n/a	n/a	●	
	ColdfireLite	39K	11K	●	●
	allports	1K	636	●	●
	headers	0	0	●	
	mcf_specific	7K	1K	●	●
	cksum.s	218	0	●	●
	iuart.c	1748	827	●	●
	osporttk.c	208	16	●	●
	tk_util.s	136	0	●	●
	iutil.c	648	124	●	●
	ifec.c	4816	811	●	●
	mip	6K	1K	●	●
	misclib	9K	4K	●	●
	mtcp	12K	2K	●	●
	net	2K	641	●	●
	tftp	0	0		
	vfs	0	0		
	project files	1K	640	●	●
	Int_handlers.c	824	173	●	●
	main.c	956	467	●	●
✓	freescale_web_server	6K	48K	●	●
	freescale_static_ffs_util...	580	522	●	●
	freescale_dynamic_http.c	1676	251	●	●
	freescale_flash_loader.c	284	45	●	●
	freescale_http.c	3248	1049	●	●
	freescale_http_server.c	644	370	●	●
	freescale_http_server.h	0	0	●	
✓	freescale_static_ffs.c	0	47479	●	●
	183 files	60K	64K		

```
//*************************************************************************
//* Static Flash File System Generator
//* Written by Eric Gregori - Chicago FAE
//*
//*************************************************************************


    const unsigned char readme_htm[] = {
    0x48,0x54,0x54,0x50,0x2F,0x31,0x2E,0x31,0x20,0x32,
    0x30,0x30,0x20,0x4F,0x4B,0x0D,0x0A,
    0x53,0x65,0x72,0x76,0x65,0x72,0x3A,0x20,0x45,0x4D,
    0x47,0x2F,0x31,0x2E,0x31,0x2E,0x30,0x0D,0x0A,
    0x43,0x61,0x63,0x68,0x65,0x2D,0x63,0x6F,0x6E,0x74,
    0x72,0x6F,0x6C,0x3A,0x20,0x6E,0x6F,0x2D,0x63,0x61,
    0x63,0x68,0x65,0x0D,0x0A,
    0x48,0x54,0x54,0x50,0x20,0x53,0x65,0x72,0x76,0x65,
    0x72,0x20,0x42,0x79,0x20,0x45,0x72,0x69,0x63,0x20,
    0x47,0x72,0x65,0x67,0x6F,0x72,0x69,0x20,0x56,0x65,
    0x72,0x73,0x69,0x6F,0x6E,0x20,0x31,0x2E,0x31,0x2E,
    0x30,0x0D,0x0A,
    0x43,0x6F,0x6E,0x74,0x65,0x6E,0x74,0x2D,0x74,0x79,
    0x70,0x65,0x3A,0x20,
    0x74,0x65,0x78,0x74,0x2F,0x68,0x74,0x6D,0x6C,
    0x0D,0x0A,
    0x43,0x6F,0x6E,0x74,0x65,0x6E,0x74,0x2D,0x6C,0x65,
    0x6E,0x67,0x74,0x68,0x3A,0x20,
    0x33,0x34,0x33,0x35,0x32,0x0D,0x0A,0x0D,0x0A,
    0x3C,0x68,0x74,0x6D,0x6C,0x3E,0x0D,0x0A,0x3C,0x68,
    0x65,0x61,0x64,0x3E,0x0D,0x0A,0x3C,0x74,0x69,0x74,
    0x6C,0x65,0x3E,0x44,0x79,0x6E,0x61,0x6D,0x69,0x63,
    0x20,0x48,0x54,0x54,0x50,0x20,0x73,0x65,0x72,0x76,
    0x65,0x72,0x20,0x77,0x69,0x74,0x68,0x20,0x73,0x69,
    0x6D,0x70,0x6C,0x65,0x20,0x46,0x6C,0x61,0x73,0x68,
    0x20,0x46,0x69,0x6C,0x65,0x20,0x53,0x79,0x73,0x74,
    0x65,0x6D,0x3C,0x2F,0x74,0x69,0x74,0x6C,0x65,0x3E,
    0x0D,0x0A,0x3C,0x2F,0x68,0x65,0x61,0x64,0x3E,0x0D,
    0x0A,0x3C,0x62,0x6F,0x64,0x79,0x3E,0x0D,0x0A,0x3C,
    0x49,0x4D,0x47,0x20,0x53,0x52,0x43,0x3D,0x22,0x43,
    0x46,0x43,0x4F,0x52,0x45,0x53,0x45,0x4D,0x42,0x4C,
    0x45,0x4D,0x2E,0x67,0x69,0x66,0x22,0x3E,0x0D,0x0A,
```

Start the Flash Programmer

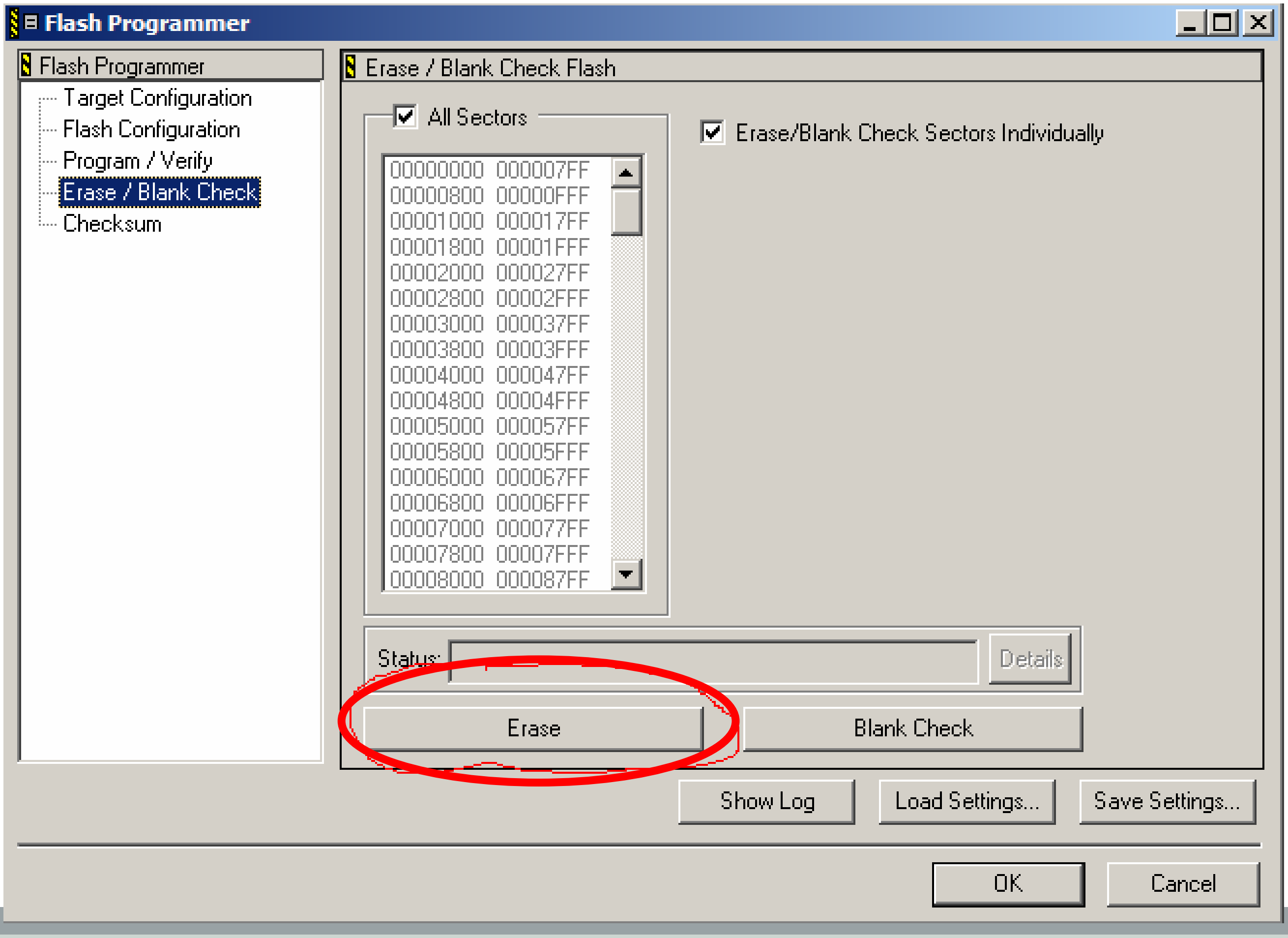
Start the Flash Programmer by selecting the tools Flash Programmer Pull Down



//******************

Erase Flash

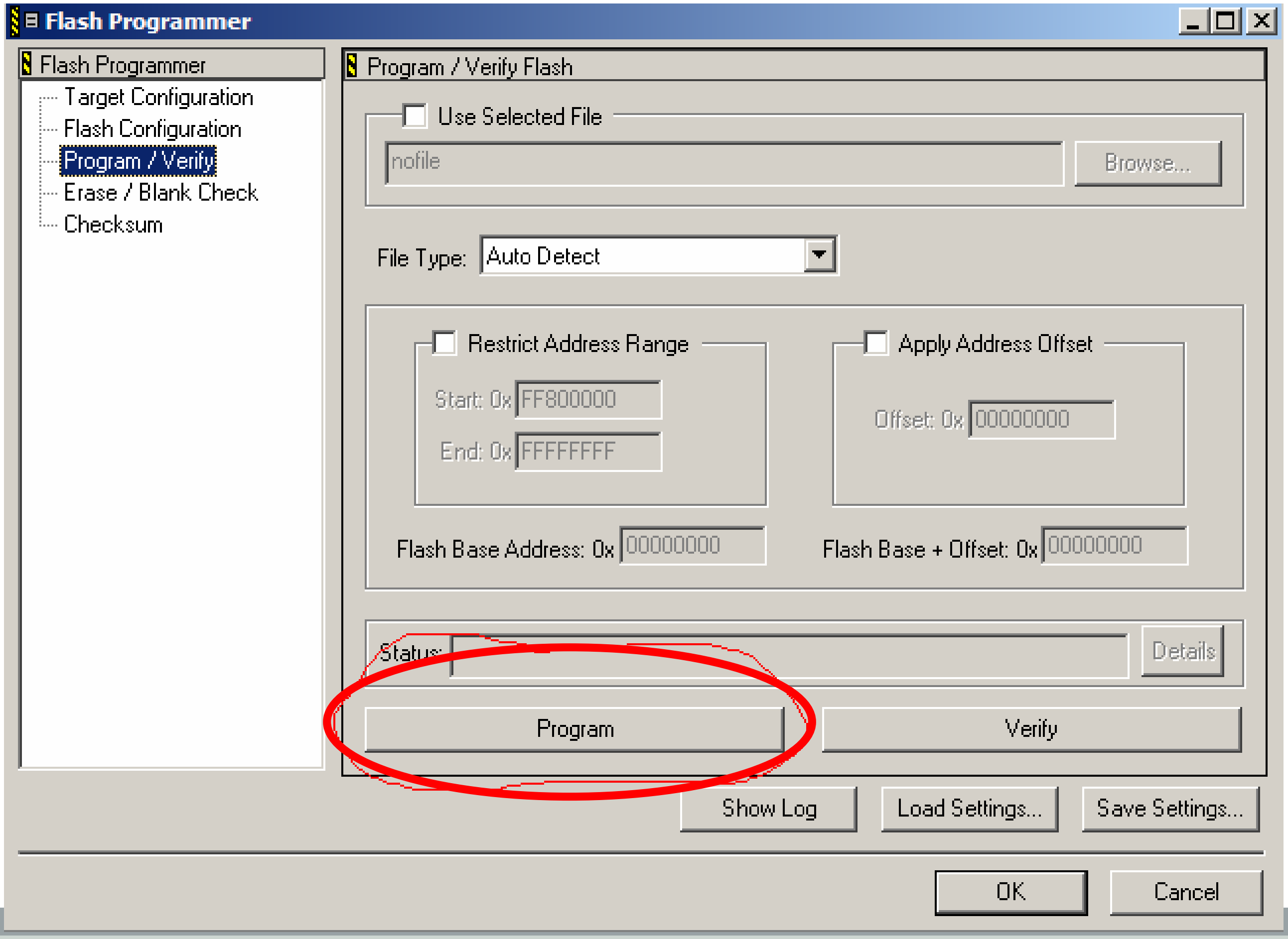
Erase Flash by selecting Erase/Blank Check, and clicking the Erase button. Watch the Status window for errors.



```
00000000  000007FF
00000800  00000FFF
00001000  000017FF
00001800  00001FFF
00002000  000027FF
00002800  00002FFF
00003000  000037FF
00003800  00003FFF
00004000  000047FF
00004800  00004FFF
00005000  000057FF
00005800  00005FFF
00006000  000067FF
00006800  00006FFF
00007000  000077FF
00007800  00007FFF
00008000  000087FF
```

Program

After the Erase is Complete, go to the Program/Verify window and click on the Program button.



Flash Programmer

- Flash Programmer
  - Target Configuration
  - Flash Configuration
  - Program / Verify
  - Erase / Blank Check
  - Checksum

Program / Verify Flash

☐ Use Selected File: nofile

File Type: Auto Detect

☐ Restrict Address Range — Start: 0x FF800000, End: 0x FFFFFFFF

☐ Apply Address Offset — Offset: 0x 00000000

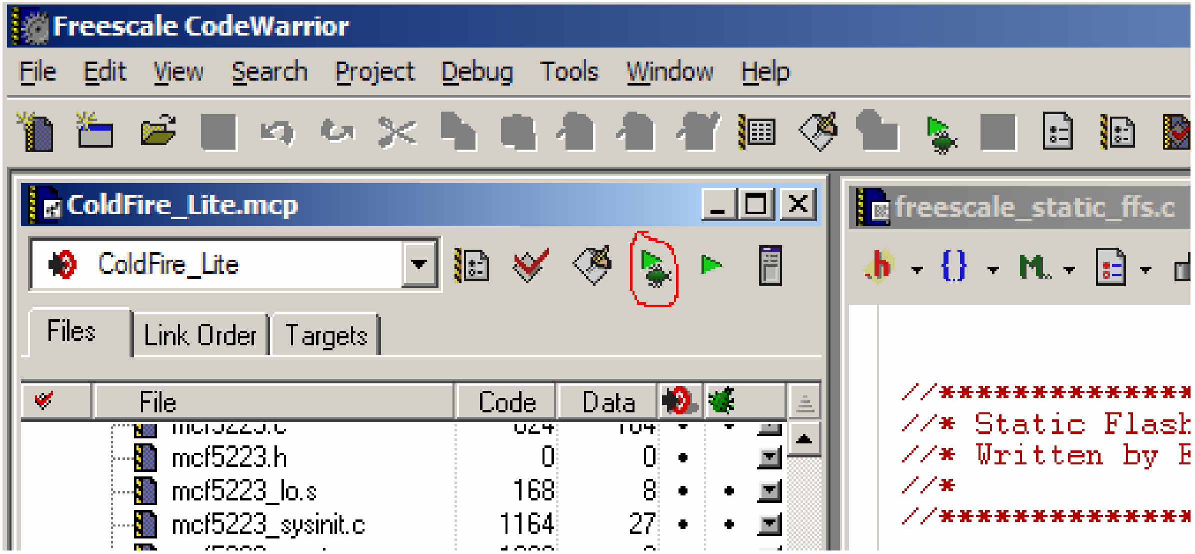
Flash Base Address: 0x 00000000

Flash Base + Offset: 0x 00000000

Status:

Run

Click on the Run icon, circled in RED below. This will execute the code in flash. If you have an external power supply, you could also disconnect the USB from the board and hit reset.

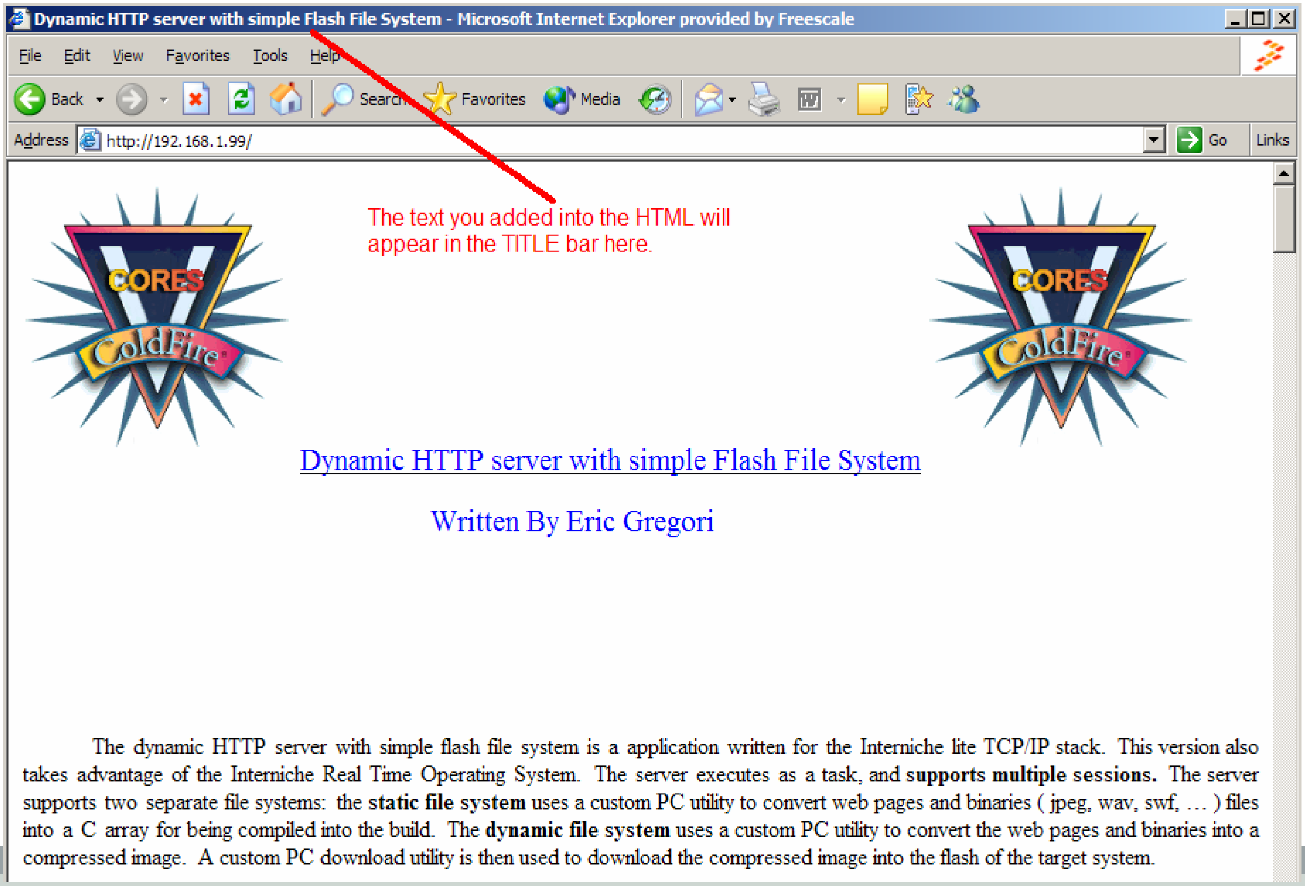


File	Code	Data		
mcf5223.h	0	0	•	
mcf5223_lo.s	168	8	•	•
mcf5223_sysinit.c	1164	27	•	•

```
//**************
//* Static Flash
//* Written by F
//*
//**************
```

Default Web Page

Open Internet Explorer, and type 192.168.1.99 (the IP address of the demo board) into the address bar. This is the default compile time web page you just loaded with the TCP/IP stack and Web Server.


Dynamic HTTP server with simple Flash File System - Microsoft Internet Explorer provided by Freescale
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
http://192.168.1.99/
Go
Links
The text you added into the HTML will appear in the TITLE bar here.
CORES
ColdFire

Dynamic HTTP server with simple Flash File System

Written By Eric Gregori

The dynamic HTTP server with simple flash file system is a application written for the Interniche lite TCP/IP stack. This version also takes advantage of the Interniche Real Time Operating System. The server executes as a task, and **supports multiple sessions.** The server supports two separate file systems: the **static file system** uses a custom PC utility to convert web pages and binaries (jpeg, wav, swf, …) files into a C array for being compiled into the build. The **dynamic file system** uses a custom PC utility to convert the web pages and binaries into a compressed image. A custom PC download utility is then used to download the compressed image into the flash of the target system.

Run Time Loadable Web Pages

Web Pages can be uploaded via Ethernet at run time.

Web Pages can be loaded over and over again. # of re-loads only limited by # of writes to flash.

Loaded Web Pages take priority over default or Compile Time Web Pages.

Loaded Web Pages are protected with a 32 character password string.

Serial Flash Support

The firmware supports a SPI based external serial flash.

Serial flash parts are available in 1 Mbyte and 4 Mbyte sizes.

When serial flash is enabled, all 256K of on-board flash is available for user firmware.

Build and Loading a Run Time Loadable Web Page

A single Batch file is used to both build and load the Web Page.

Within the Batch file are calls to two executable.

The first executable: emg_dynamic_ffs.exe

- Compresses the Web Pages into a binary, and adds a File Allocation Table (FAT) to the top of the file. The firmware in the Web Server uses the FAT to reference the data in the file from within the binary image.

Emg_dynamic_FFS.exe

```
Emg_dynamic_ffs filelist.txt output_file.ffs
```

Where:

Filelist.txt is a text file containing the list of files to compress. Each file should be on its own line, and the first file is the default.

Comments can be added using a ‘*’ as the first character in a line.

Output_file.ffs is the file generated containing all the files in the filelist compressed together, along with File Allocation Table used to reference the files from the Web Server.

Emg_web_uploader.exe

Emg_web_uploader ip_address filename.ffs key_string

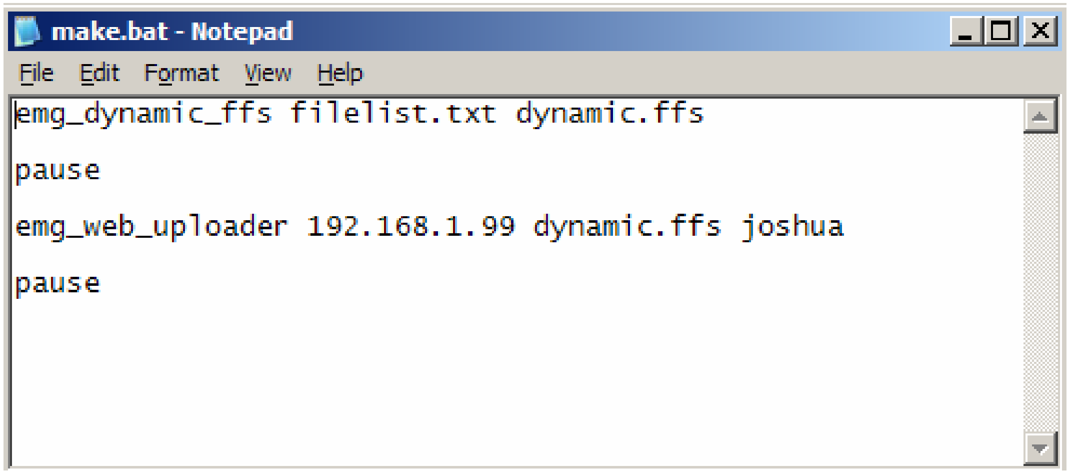
Where:

Ip_address is the ip address of the hardware (192.168.1.99) in examples.

Filename.ffs is the file generated by the emg_dynamic_ffs utility.

Key_string is the 32 character key used to unlock the flash file system (joshua) in examples.

The make.bat File



```
emg_dynamic_ffs filelist.txt dynamic.ffs

pause

emg_web_uploader 192.168.1.99 dynamic.ffs joshua

pause
```

The filelist.txt file lists the files that will be included in the FFS.

Dynamic.ffs is the binary image containing all the files and the FAT.

Pause is a DOS command to prompt the user to hit any key.

192.168.1.99 is the IP address of the hardware for these examples.

Joshua is the key string for these examples.

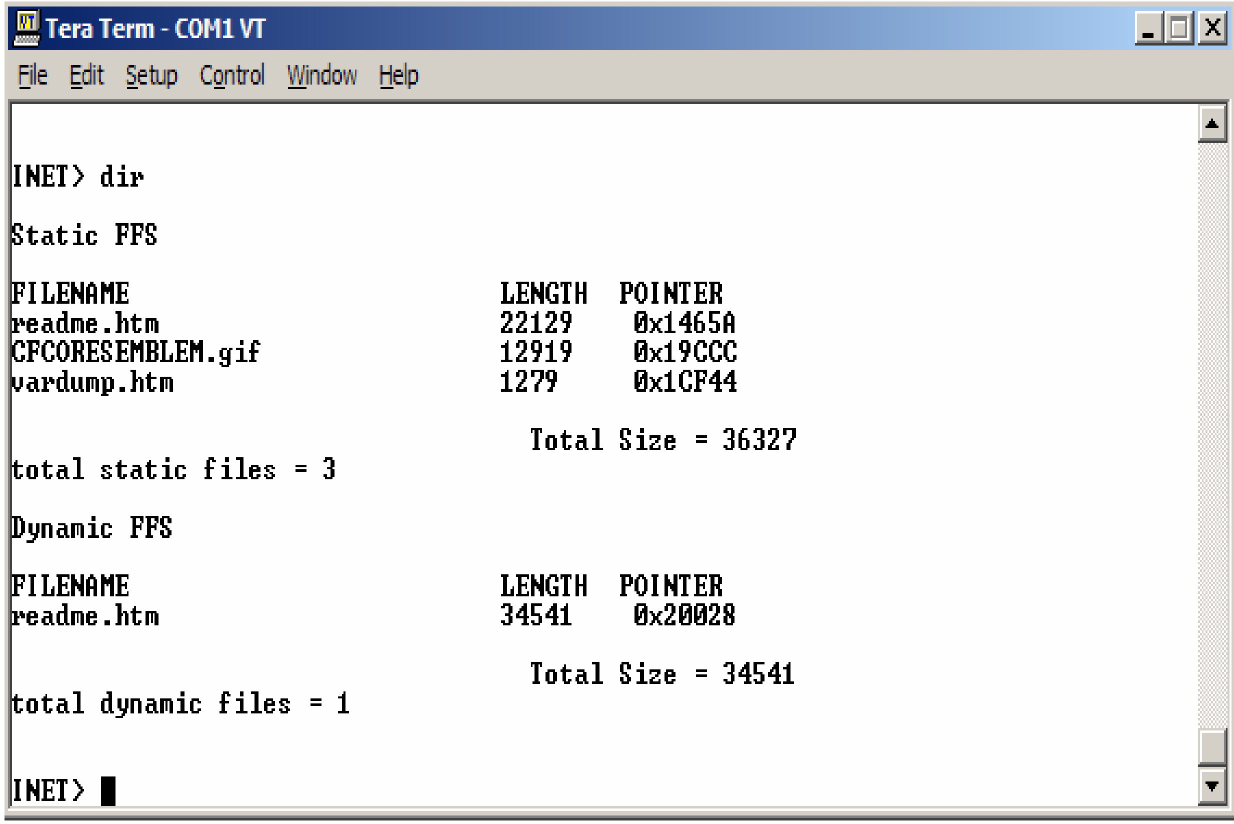
Demo Board Directory

Let take a look at the contents in the directory of the demo board

Notice, the static file system (compile time) still contains files.

When the dynamic (run time) file system is loaded with a binary image, it takes priority over the static file system.

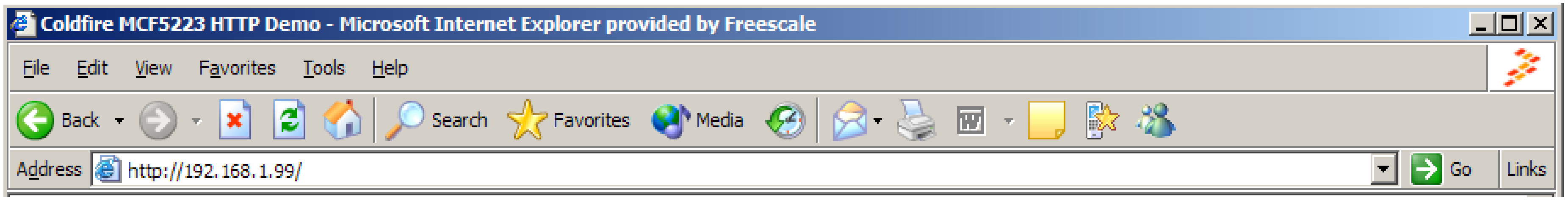
Other files in the static FFS are still available.



```
INET> dir

Static FFS

FILENAME                         LENGTH  POINTER
readme.htm                       22129    0x1465A
CFCORESEMBLEM.gif                12919    0x19CCC
vardump.htm                      1279     0x1CF44

                                   Total Size = 36327
total static files = 3

Dynamic FFS

FILENAME                         LENGTH  POINTER
readme.htm                       34541    0x20028

                                   Total Size = 34541
total dynamic files = 1

INET>
```

Web Server Defaults

Notice what we entered at the address bar. No filename is specified. When no filename is specified the Web Server defaults to the first file listed in the file system.


Coldfire MCF5223 HTTP Demo - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links

http://192.168.1.99/

* emg dynamic web page description file
* The files listed below will be concantenated into a
* single compressed downloadable image.

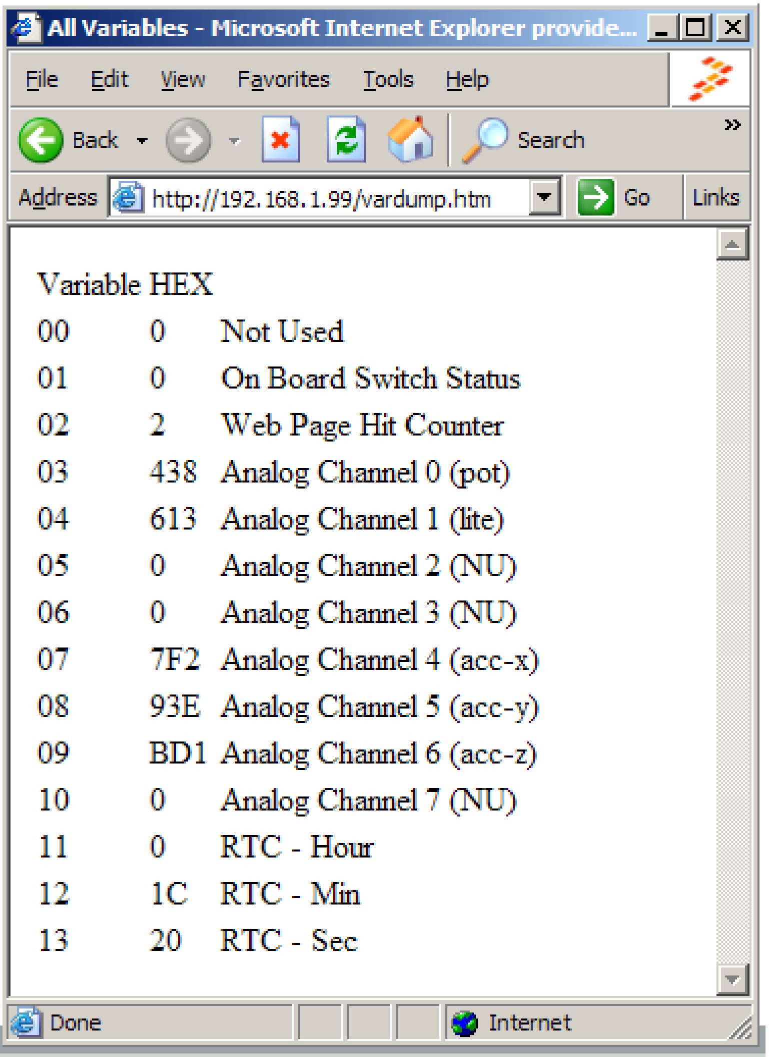
* The first file in the list is the default file

Readme.htm.htm ← This is the file that is loaded by default.
CFCORESEMBLEM.gif
vardump.htm

Going Direct to a File Using the Browser

To go directly to a file in the FFS from the browser, just include the name of the file after the '/' in the IP address.

Notice Vardump.htm is in the static file system, but is still available after loading a dynamic FFS.


All Variables - Microsoft Internet Explorer provide...
File Edit View Favorites Tools Help
Back Search
Address http://192.168.1.99/vardump.htm Go Links
Done Internet

Variable	HEX	
00	0	Not Used
01	0	On Board Switch Status
02	2	Web Page Hit Counter
03	438	Analog Channel 0 (pot)
04	613	Analog Channel 1 (lite)
05	0	Analog Channel 2 (NU)
06	0	Analog Channel 3 (NU)
07	7F2	Analog Channel 4 (acc-x)
08	93E	Analog Channel 5 (acc-y)
09	BD1	Analog Channel 6 (acc-z)
10	0	Analog Channel 7 (NU)
11	0	RTC - Hour
12	1C	RTC - Min
13	20	RTC - Sec

The WEB Server Firmware - Processing a FORM Submit

The Web Server detects the form by the ‘?’ in the filename.

The FORM is then parsed into the two parts, the NAME and the VALUE.

The NAME is on the left of the ‘=‘ sign, the VALUE on the right.

The Name is used to call the function “LED”, and pass it the VALUE.*

Dynamic HTML Tokens

Dynamic HTML Tokens allow variable content like sensor data to be inserted into web pages, no programming required.

Just insert the token ~IIF; into your HTML, and the token will be replaced with the data referenced by II.

Conditional tokens take the idea one step further, by allowing whole HTML strings to be replaced based a data comparison to a constant.

The REPLACE Token ~IIF;

Where:

II = The decimal variable index to read the data.

- The variable array contains 32 longwords (can be as high as 99).

F = The format to display the data (D = Decimal, H = Hex).

Example:

```
<HTML>
<HEAD>
<TITLE>This text will appear at the top of the web browser, the navigation bar</TITLE>
</HEAD>
<BODY>
<CENTER>You have opened this page ~02D; times</CENTER>
</BODY>
</HTML>
```

The Variable index 02 is the web page hit counter.

The CONDITIONAL Token ^II>C|true|false|;

Where:

II	=	The decimal variable index to read the data. The variable array contains 32 longwords (can be as high as 99)
C	=	Hex value for comparison.
>	=	Variable value greater then C
=	=	Variable value equal to C
&	=	Variable value and C
!	=	Variable not equal to C

“true” = ascii string to replace if condition is true
“false” = ascii string to replace if condition is false

The Variable Array

	Parameter
00	Available to user
01	On Board Switch Status
02	Web Page Hit Counter
03	Analog Channel 0 (pot)
04	Analog Channel 1 (lite)
05	Analog Channel 2 (NU)
06	Analog Channel 3 (NU)
07	Analog Channel 4 (acc-x)
08	Analog Channel 5 (acc-y)
09	Analog Channel 6 (acc-z)
10	Analog Channel 7 (NU)
11	RTC - Hour
12	RTC - Min
13	RTC - Sec
14	Available to user
15	Available to user
16	Available to user
17	Available to user
18	Available to user
19	Available to user
20	Available to user
21	Available to user
22	Available to user
23	Available to user
24	Available to user
25	Available to user
26	Available to user
27	Available to user
28	Available to user
29	Available to user
30	Available to user
31	Available to user

The Serial Interface

Notice the “Available To User” entries in the variable array.

You can modify the ‘C’ code for the Web Server to assign any 32 bit value you want to a available position in the variable array.

Or, you can use the serial interface to modify the variable in the array.

The serial interface method is designed for interfacing to other embedded systems.

The serial port supports autobaud, so it will automatically sync to the baud of your embedded device.

Using the Serial Interface- The ‘VAR’ command

```
INET> var

Dynamic HTML variable dump
Variable 0 = 12345678   BC614E
Variable 1 = 0   0
Variable 2 = 1035   40B
Variable 3 = 2202   89A
Variable 4 = 2205   89D
Variable 5 = 0   0
Variable 6 = 0   0
Variable 7 = 2435   983
Variable 8 = 387   183
Variable 9 = 3125   C35
Variable 10 = 0   0
Variable 11 = 23   17
Variable 12 = 26   1A
Variable 13 = 56   38
Variable 14 = 99   63

INET>
```

VAR Command Parameters

var – Dumps the contents of the array to the serial port.

Var 14 – Dumps the contents of variable index 14.

Var 14, 12345678 – Assigns 12345678 decimal to variable index 14.

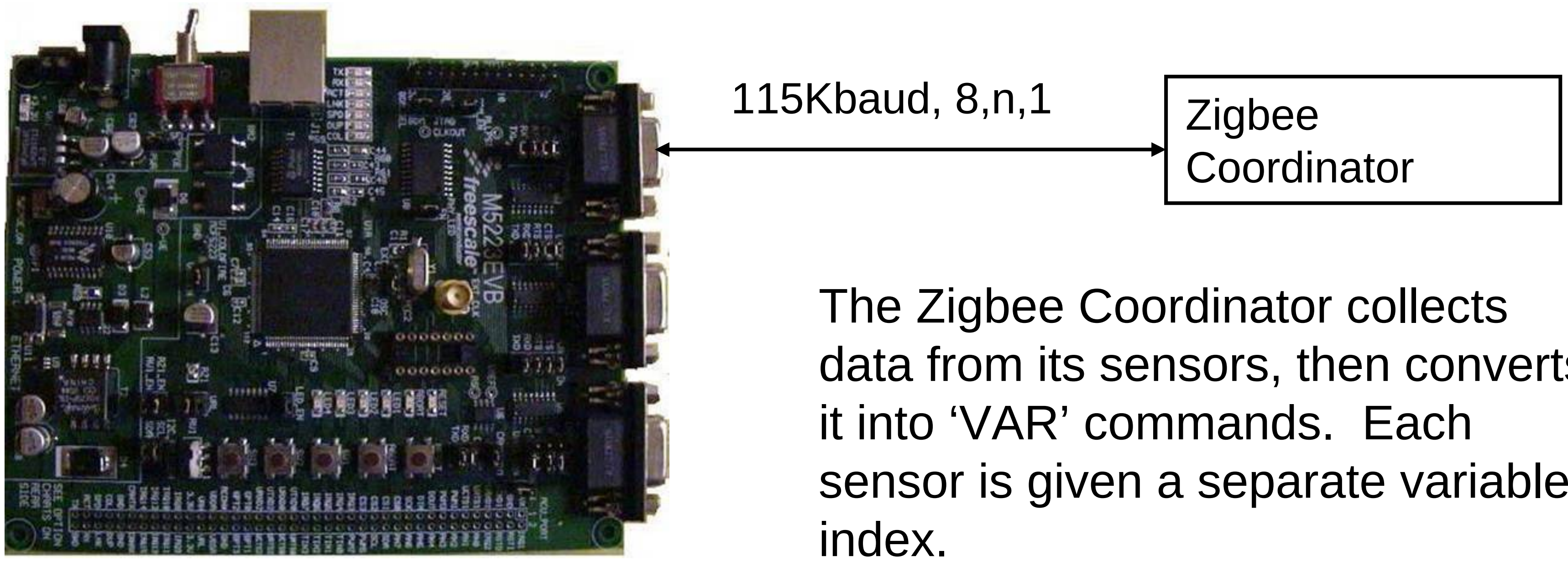
The ‘var x’ Command

```
INET> var 0

Variable 0 = 12345678   BC614E

INET> var 2

Variable 2 = 1195   4AB

INET> var 3

Variable 3 = 2202   89A

INET> var 4

Variable 4 = 2275   8E3

INET>
```

Assigning a Variable with the ‘var’ Command var II, decimal_data

```
INET> var 14, 100


INET> var 14

Variable 14 = 100   64

INET> var 14,250


INET> var 14

Variable 14 = 250   FA

INET> var 14, 900


INET> var 14

Variable 14 = 900   384

INET>
```

How to Use the VAR Command



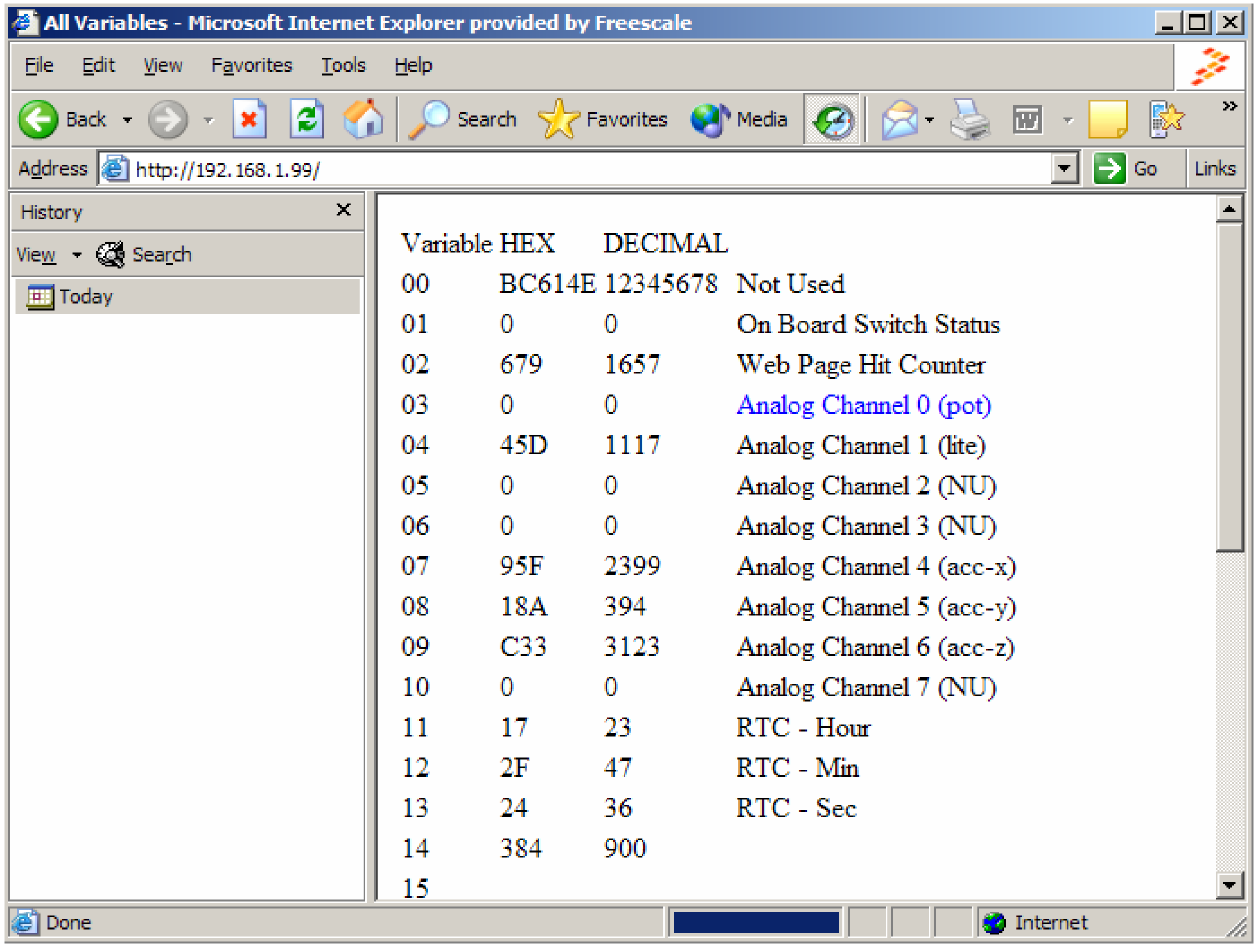
The Zigbee Coordinator collects data from its sensors, then converts it into 'VAR' commands. Each sensor is given a separate variable index.

The 'VAR' command is terminated with a CR, the INET> prompt provides software handshaking.

The HTML Code

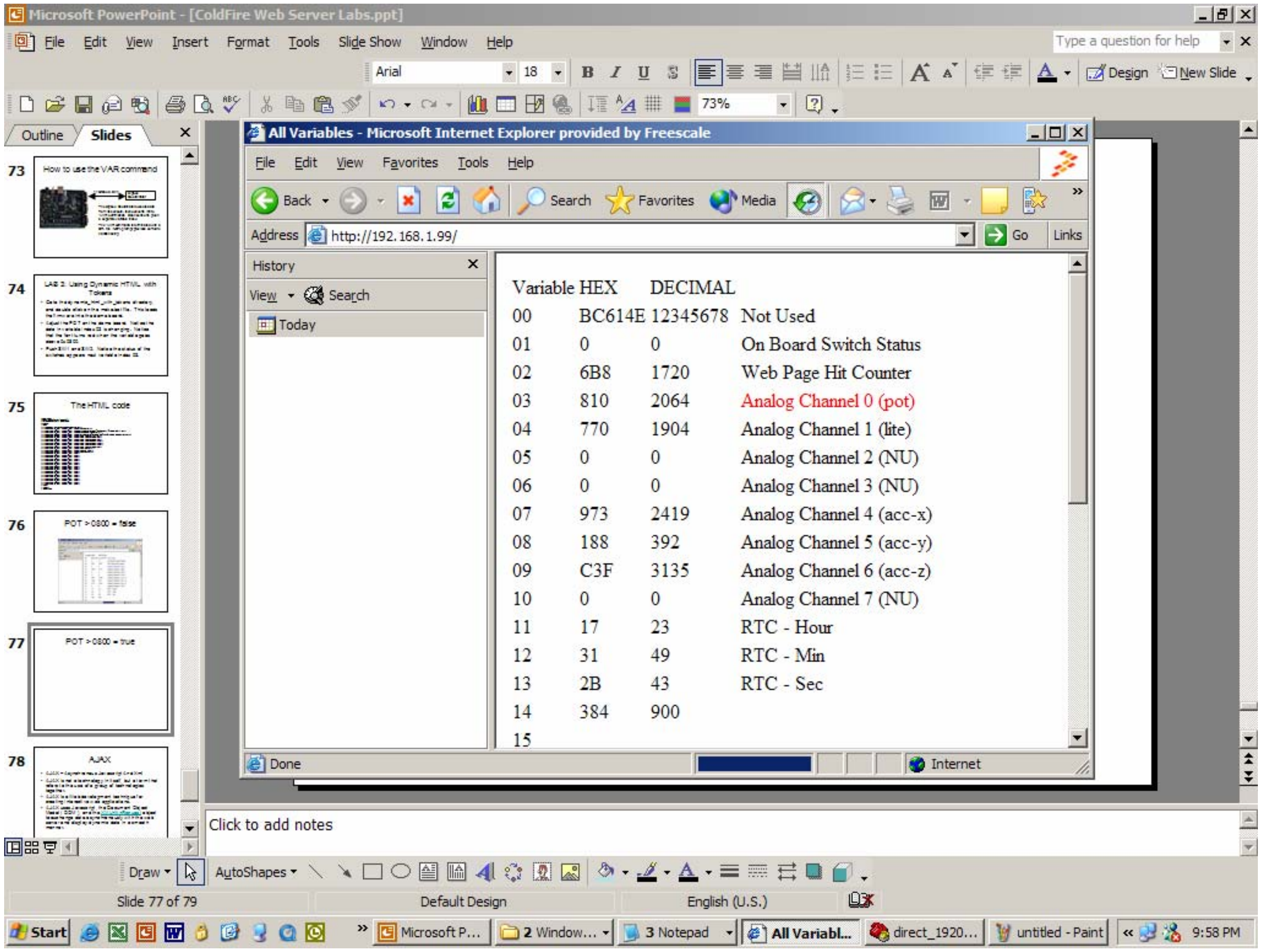
```
<html><head>
<meta http-equiv="refresh" content="1">
<title>All Variables</title></head><body>
<TABLE>
<tbody>
<tr><td>Variable</td><td>HEX</td><td>DECIMAL</td></tr>
<tr><td>00</td><td>~00H    ;</td><td>~00D    ;</td><td>Not Used</td></tr>
<tr><td>01</td><td>~01H    ;</td><td>~01D    ;</td><td>On Board Switch Status ^01&0001|SW1||;^01&0008|SW2||;</td></tr>
<tr><td>02</td><td>~02H    ;</td><td>~02D    ;</td><td>Web Page Hit Counter</td></tr>
<tr><td>03</td><td>~03H    ;</td><td>~03D    ;</td><td><FONT COLOR=^03>0800|"RED"|"BLUE"|;>Analog Channel 0 (pot)</td></tr>
<tr><td>04</td><td>~04H    ;</td><td>~04D    ;</td><td>Analog Channel 1 (lite)</td></tr>
<tr><td>05</td><td>~05H    ;</td><td>~05D    ;</td><td>Analog Channel 2 (NU)</td></tr>
<tr><td>06</td><td>~06H    ;</td><td>~06D    ;</td><td>Analog Channel 3 (NU)</td></tr>
<tr><td>07</td><td>~07H    ;</td><td>~07D    ;</td><td>Analog Channel 4 (acc-x)</td></tr>
<tr><td>08</td><td>~08H    ;</td><td>~08D    ;</td><td>Analog Channel 5 (acc-y)</td></tr>
<tr><td>09</td><td>~09H    ;</td><td>~09D    ;</td><td>Analog Channel 6 (acc-z)</td></tr>
<tr><td>10</td><td>~10H    ;</td><td>~10D    ;</td><td>Analog Channel 7 (NU)</td></tr>
<tr><td>11</td><td>~11H    ;</td><td>~11D    ;</td><td>RTC - Hour</td></tr>
<tr><td>12</td><td>~12H    ;</td><td>~12D    ;</td><td>RTC - Min </td></tr>
<tr><td>13</td><td>~13H    ;</td><td>~13D    ;</td><td>RTC - Sec </td></tr>
<tr><td>14</td><td>~14H    ;</td><td>~14D    ;</td></tr>
<tr><td>15</td><td>~15H    ;</td><td>~15D    ;</td></tr>
<tr><td>16</td><td>~16H    ;</td><td>~16D    ;</td></tr>
<tr><td>17</td><td>~17H    ;</td><td>~17D    ;</td></tr>
<tr><td>18</td><td>~18H    ;</td><td>~18D    ;</td></tr>
<tr><td>19</td><td>~19H    ;</td><td>~19D    ;</td></tr>
<tr><td>20</td><td>~20H    ;</td><td>~20D    ;</td></tr>
<tr><td>21</td><td>~21H    ;</td><td>~21D    ;</td></tr>
<tr><td>22</td><td>~22H    ;</td><td>~22D    ;</td></tr>
<tr><td>23</td><td>~23H    ;</td><td>~23D    ;</td></tr>
<tr><td>24</td><td>~24H    ;</td><td>~24D    ;</td></tr>
<tr><td>25</td><td>~25H    ;</td><td>~25D    ;</td></tr>
<tr><td>26</td><td>~26H    ;</td><td>~26D    ;</td></tr>
<tr><td>27</td><td>~27H    ;</td><td>~27D    ;</td></tr>
<tr><td>28</td><td>~28H    ;</td><td>~28D    ;</td></tr>
<tr><td>29</td><td>~29H    ;</td><td>~29D    ;</td></tr>
<tr><td>30</td><td>~30H    ;</td><td>~30D    ;</td></tr>
<tr><td>31</td><td>~31H    ;</td><td>~31D    ;</td></tr>
</tbody>
</TABLE>
</body></html>
```

POT > 0800 = false


All Variables - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.1.99/
Go Links
History
View Search
Today
Done
Internet

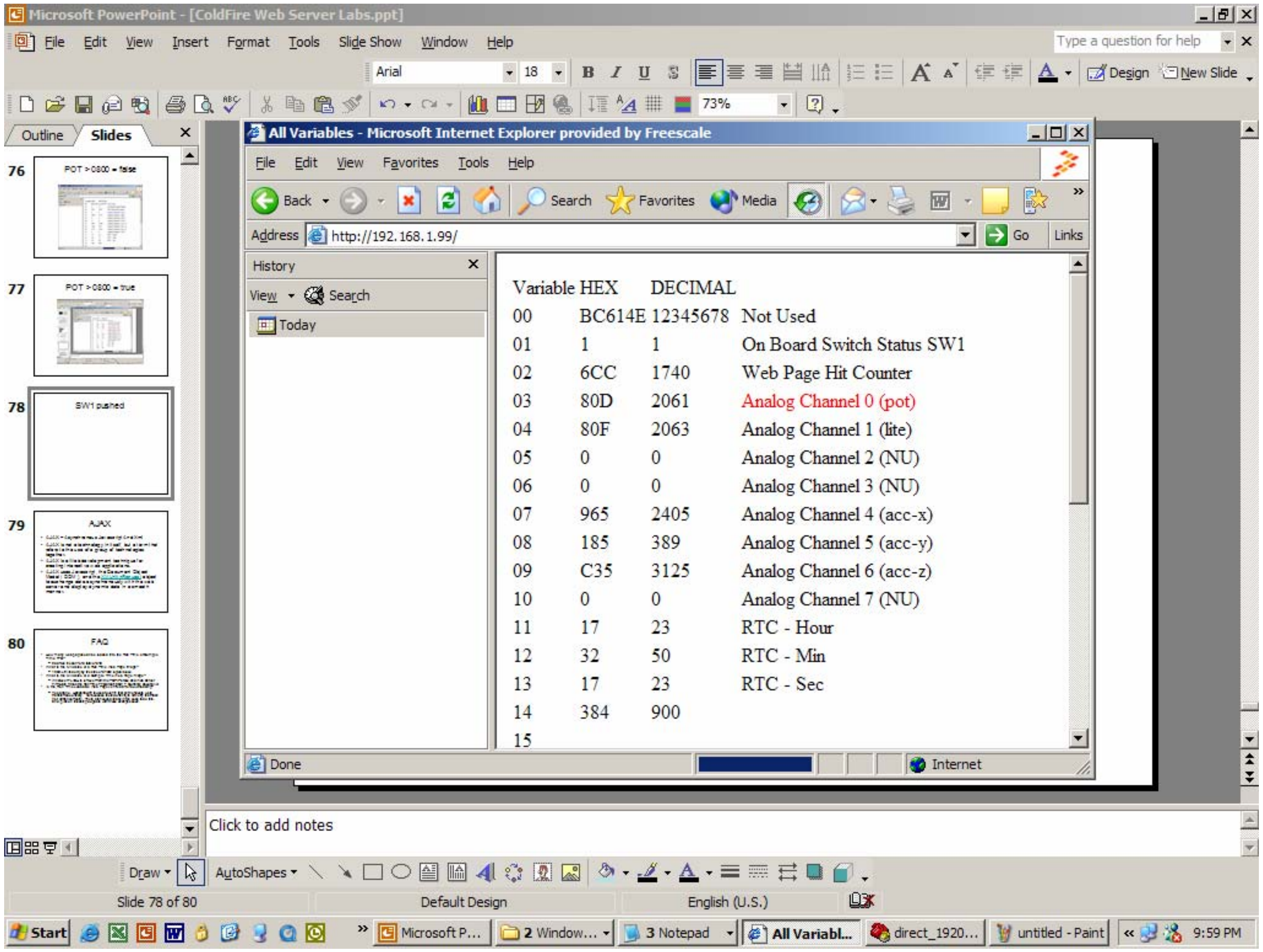
```
Variable HEX     DECIMAL
00       BC614E  12345678  Not Used
01       0       0         On Board Switch Status
02       679     1657      Web Page Hit Counter
03       0       0         Analog Channel 0 (pot)
04       45D     1117      Analog Channel 1 (lite)
05       0       0         Analog Channel 2 (NU)
06       0       0         Analog Channel 3 (NU)
07       95F     2399      Analog Channel 4 (acc-x)
08       18A     394       Analog Channel 5 (acc-y)
09       C33     3123      Analog Channel 6 (acc-z)
10       0       0         Analog Channel 7 (NU)
11       17      23        RTC - Hour
12       2F      47        RTC - Min
13       24      36        RTC - Sec
14       384     900
15
```

POT > 0800 = true



```
Variable HEX      DECIMAL
00       BC614E   12345678  Not Used
01       0        0         On Board Switch Status
02       6B8      1720      Web Page Hit Counter
03       810      2064      Analog Channel 0 (pot)
04       770      1904      Analog Channel 1 (lite)
05       0        0         Analog Channel 2 (NU)
06       0        0         Analog Channel 3 (NU)
07       973      2419      Analog Channel 4 (acc-x)
08       188      392       Analog Channel 5 (acc-y)
09       C3F      3135      Analog Channel 6 (acc-z)
10       0        0         Analog Channel 7 (NU)
11       17       23        RTC - Hour
12       31       49        RTC - Min
13       2B       43        RTC - Sec
14       384      900
15
```

SW1 Pushed



http://192.168.1.99/

Variable	HEX	DECIMAL	
00	BC614E	12345678	Not Used
01	1	1	On Board Switch Status SW1
02	6CC	1740	Web Page Hit Counter
03	80D	2061	Analog Channel 0 (pot)
04	80F	2063	Analog Channel 1 (lite)
05	0	0	Analog Channel 2 (NU)
06	0	0	Analog Channel 3 (NU)
07	965	2405	Analog Channel 4 (acc-x)
08	185	389	Analog Channel 5 (acc-y)
09	C35	3125	Analog Channel 6 (acc-z)
10	0	0	Analog Channel 7 (NU)
11	17	23	RTC - Hour
12	32	50	RTC - Min
13	17	23	RTC - Sec
14	384	900	
15			

Browser Update

Notice how the last lab updated itself in the browser

The <meta http-equiv="refresh" content="1"> HTML tag causes the page to automatically reload.

The “1” is the number of seconds to wait before reloading the page.

This is the old method of automatically updating a web page.

Notice its not very efficient, the whole page is reloaded even though only a few values change.

Notice the page flickers.

These limitations are addressed in WEB2.0.

WEB 2.0

Web 2.0 generally refers to a second generation of services available on the World Wide Web that gives users an experience closer to a desktop application than the traditional static web pages.

The traditional world wide web was designed to present static information.

Web 2.0 is designed to be interactive.

AJAX - A Key Component of Web 2.0

AJAX – Asynchronous Javascript And XML

AJAX is not a technology in itself, but a term that refers to the use of a group of technologies together.

AJAX is a Web development technique for creating interactive web applications.

AJAX uses Javascript, the Document Object Model (DOM), and the XMLHttpRequest object to exchange data asynchronously with the web server and display dynamic data in a smooth manner.

Javascript

Javascript is a prototype-based scripting language with a syntax loosely based on ‘C’.

Javascript is embedded as ascii source in web pages.

The web browser interprets the Javascript within the <HTML> tags.

Since the browser actually runs the Javascript, all the web server has to do is serve it up.

Including Javascript in your we pages is easy.

Simple “Hello World” in Javascript

```
<html>
<head>
<title>Simple Javascript</title>
</head>
<script language="JavaScript">
document.write(“Hello World”);
</script>
</html>
```

Document Object Model (DOM)

Javascript would be relatively useless if it could not alter the web page.

Of course, Javascript can alter the web page using the DOM.

The DOM makes everything on a web page a object accessible by Javascript.

Javascript accesses the object using the object ID.

Javascript + DOM

Remember the marquee in the web page from the last lab

<marquee width="800" scrollamount=8>Time Since Last Reset:

~11D;~12D;~13D;</marquee>

We modify it slightly by adding the id element

<marquee id="scroller" width="800" scrollamount=8>Time Since Last Reset:

~11D;~12D;~13D;</marquee>

Now, we can alter the marquee from Javascript.

Javascript Runs in the Background

The time in the web page automatically updates.

The time is actually being read from the ***ColdFire***® evaluation board Real Time Clock.

Javascript uses the XMLHttpRequest function to request data from the web server, without effecting the viewable page.

Javascript and Internet Explorer

Internet Explorer has an issue terminating Javascript.

Between the Javascript labs, you should close and re-open Internet Explorer.

LAB 9: Accelerameter Example

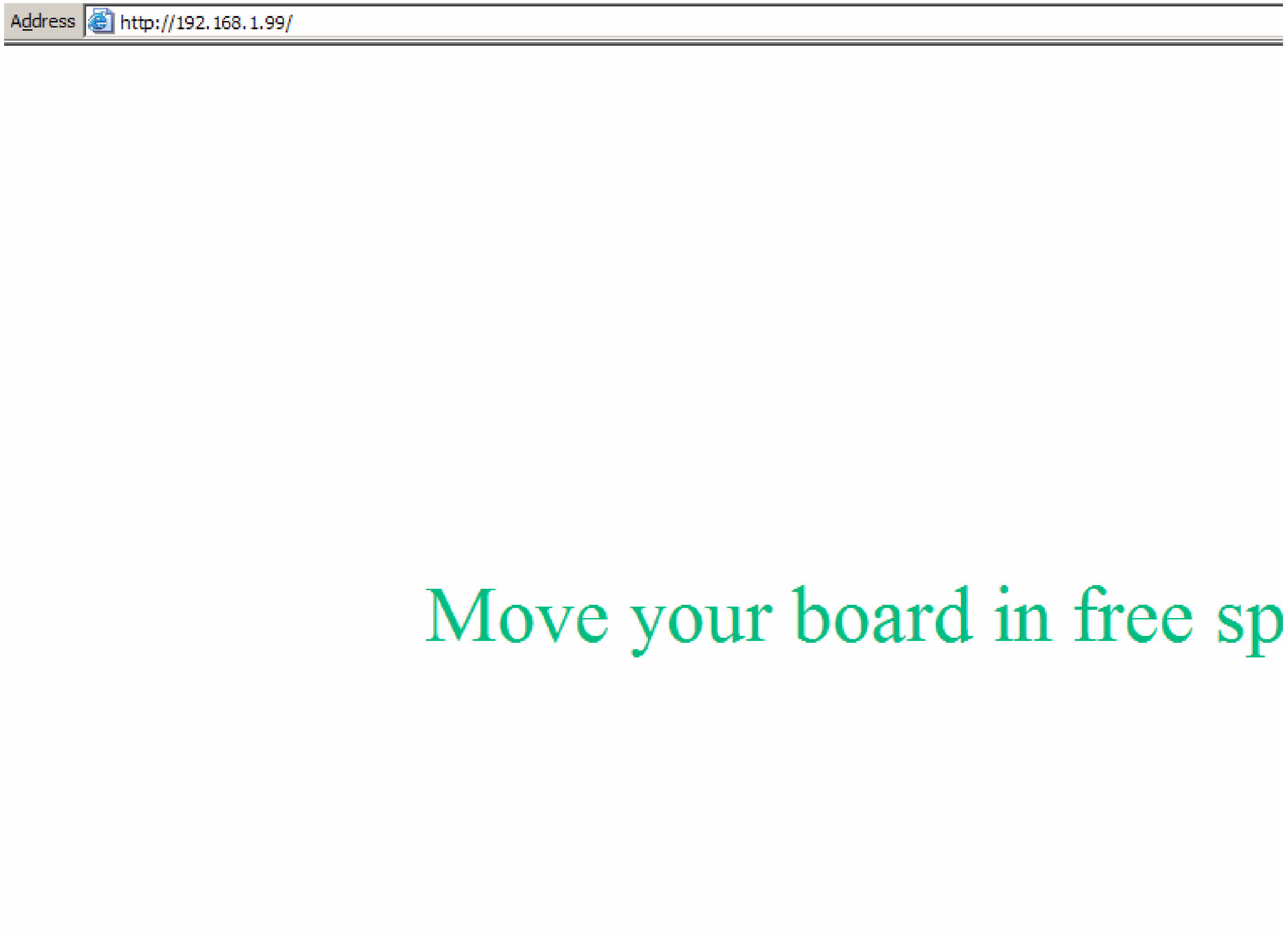
- Goto the LAB9_?????? Directory.
- Double Click the make.bat to load the LAB into the ColdFire.

The 52233DEMO board has a 3-axis accelerameter. This device outputs 3 analog voltages representing the x, y, and z planes.

The ColdFire has 2 separate 4 channel 12 bit A/D converters.

3 channels are used here to read the X, y, and z planes, then the A/D values are stored in VAR array locations 7, 8, and 9.

LAB 9: Accelerameter Example

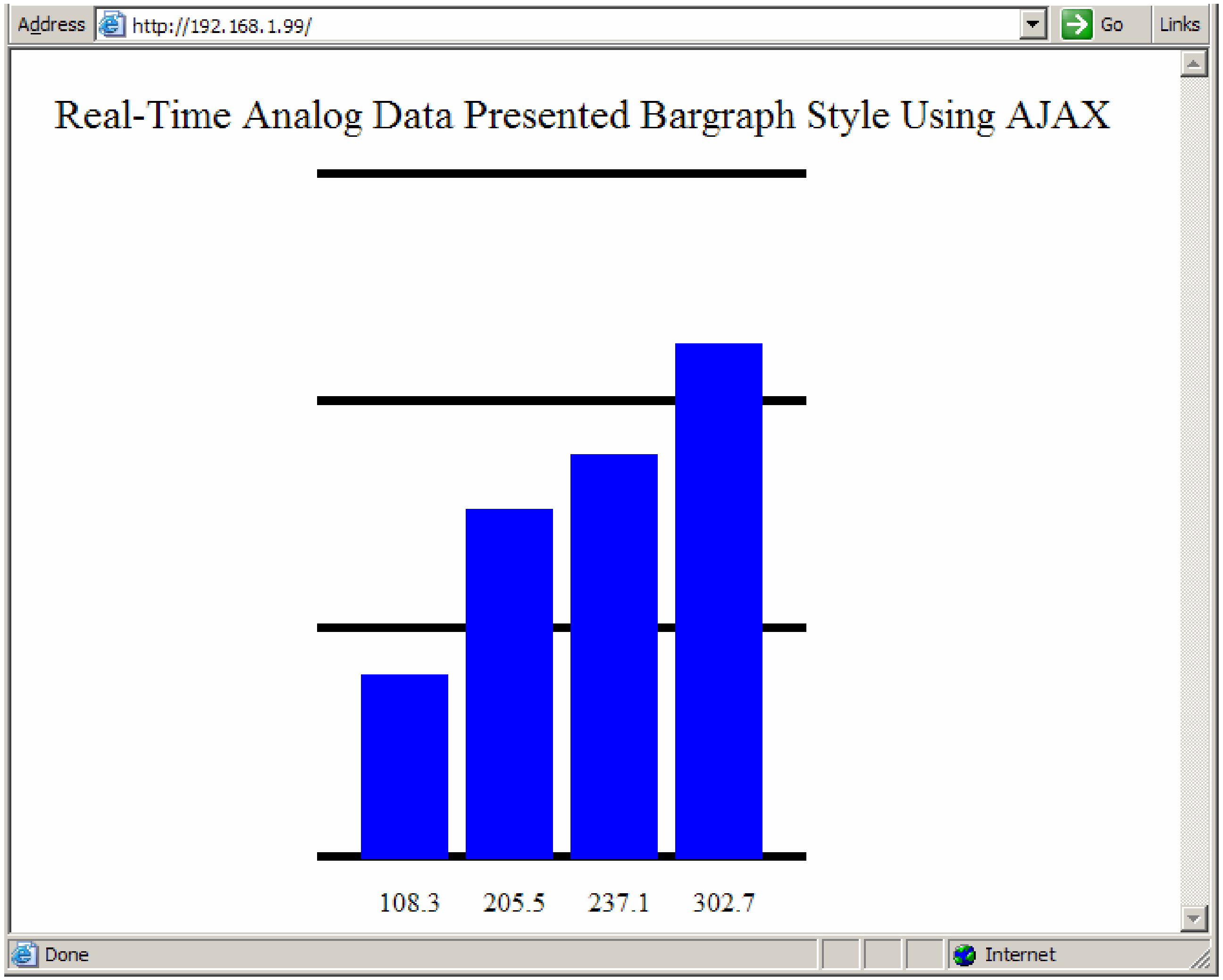

Address http://192.168.1.99/

Move your board in free sp

LAB 10: Monitoring Analog Data

- Goto the LAB10_?????? Directory.
- Double Click the make.bat to load the LAB into the ColdFire.

LAB 10: Monitoring Analog Data



Real-Time Analog Data Presented Bargraph Style Using AJAX

HTML and Javascript

Notice the image has been given an id of bargraph

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Freescale MCF5223x</title>
</head>

<body>

<IMG SRC="avtlogo.gif" id="bargraph" BORDER=0 WIDTH=549
HEIGHT=470 >

</body>
```

HTML and Javascript

The Javascript assigns the height of the bargraph object to the pot_value/10

```
<script language="JavaScript">
/////////////////////////////////////////////////////////////////////////////////////
// Javascript for 5223EVB demo Written by Eric Gregori
//
// The script communicates with the board using a AJAX technique.
/////////////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////////////
// Variables global to script
/////////////////////////////////////////////////////////////////////////////////////
var pot_value


/////////////////////////////////////////////////////////////////////////////////////
// Parse input file
/////////////////////////////////////////////////////////////////////////////////////
function parse_vars( data )
{
		var parsed = data.split( "\n" );

		pot_value  = parsed[0]

		bargraph.height = pot_value/10
}
```

HTML and Javascript

The Javascript request the data from the server using http_request.open('GET', url, true);

```
//////////////////////////////////////////////////////////////////////////////////////
// Request input file
//////////////////////////////////////////////////////////////////////////////////////
function makeRequest(url)
{
                        var http_request = false;

     if (window.XMLHttpRequest)
                        { // Mozilla, Safari,...
                        http_request = new XMLHttpRequest();
                        if (http_request.overrideMimeType)
                                                {
                        http_request.overrideMimeType('text/xml');
                        }
     }
                        else if (window.ActiveXObject)
                        { // IE
                                                try
                                                {
                        http_request = new ActiveXObject("Msxml2.XMLHTTP");
                        }
                                                catch (e)
                                                {
                        try
                                                                        {
                                                http_request = new ActiveXObject("Microsoft.XMLHTTP");
                        }
                                                                        catch (e) {}
                        }
     }

     if (!http_request)
                        {
                        alert('Giving up :( Cannot create an XMLHTTP instance');
                        return false;
     }

     http_request.onreadystatechange = function() { alertContents(http_request); };
     http_request.open('GET', url, true);
     http_request.send(null);
}
```

HTML and Javascript

The javascript request the data from the server by requesting the file pot_data.txt
This request is done every 200ms (setTimeout).

```
//////////////////////////////////////////////////////////////////////////////////////////
// Handle file request response
//////////////////////////////////////////////////////////////////////////////////////////
function alertContents(http_request)
{
                        if (http_request.readyState == 4)
                        {
                                                if (http_request.status == 200)
                                                {
                        parse_vars(http_request.responseText);
                        }
                                                else
                                                {
//                      alert('There was a problem with the request.');
//                                                                      alert( http_request.status );
                        }
       }
}

//////////////////////////////////////////////////////////////////////////////////////////
// Infinite loop with delay
//////////////////////////////////////////////////////////////////////////////////////////
function loop()
{
                        makeRequest("pot_data.txt");
                        setTimeout("loop()",200);
}

//////////////////////////////////////////////////////////////////////////////////////////
// Run
//////////////////////////////////////////////////////////////////////////////////////////
window.onload=loop;


</script>
</html>
```

LAB 11: Back to Real Work

AJAX can be used for more than fun and games.

In an embedded environment sometimes it would be nice to present real-time changing data in a graphic manner.

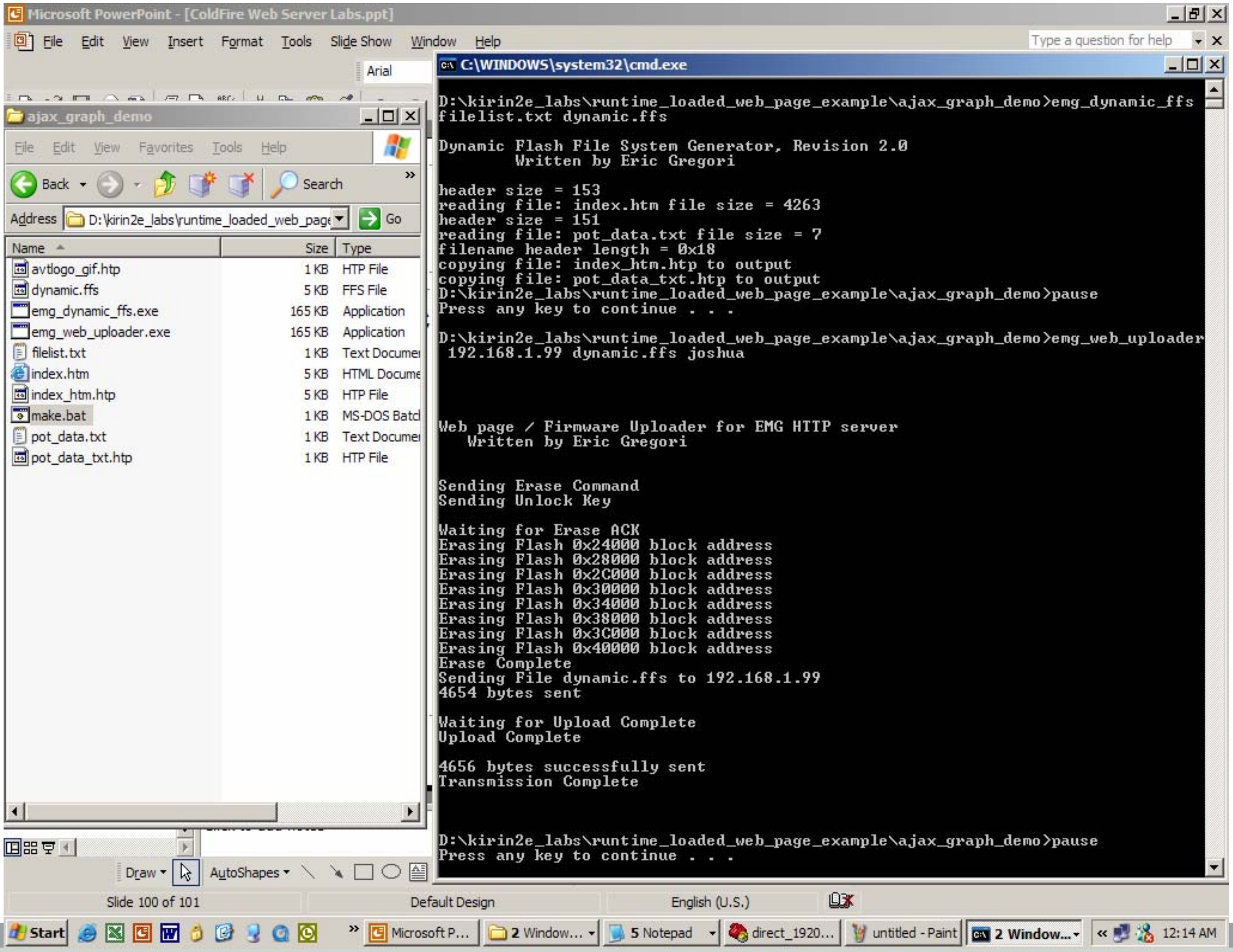
Go to the ajax_graph_demo directory.

Close the web browser (internet explorer).

Double click the make.bat file.

Open Internet Explorer, and type 192.168.1.99 in the address bar.

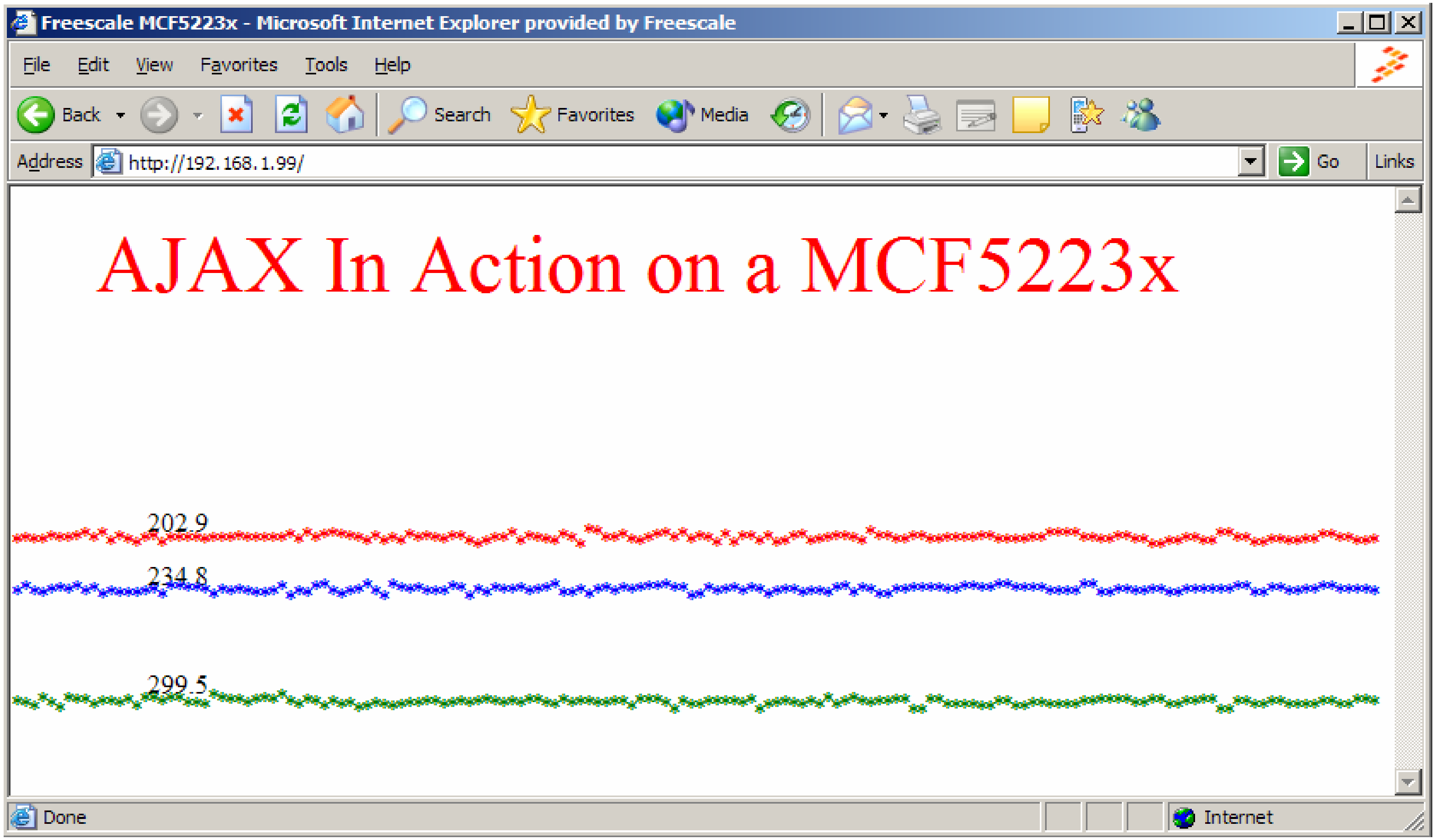
Build and Load ajax_graph_demo



Name	Size	Type
avtlogo_gif.htp	1 KB	HTP File
dynamic.ffs	5 KB	FFS File
emg_dynamic_ffs.exe	165 KB	Application
emg_web_uploader.exe	165 KB	Application
filelist.txt	1 KB	Text Docume
index.htm	5 KB	HTML Docume
index_htm.htp	5 KB	HTP File
make.bat	1 KB	MS-DOS Batd
pot_data.txt	1 KB	Text Docume
pot_data_txt.htp	1 KB	HTP File

```
D:\kirin2e_labs\runtime_loaded_web_page_example\ajax_graph_demo>emg_dynamic_ffs
filelist.txt dynamic.ffs

Dynamic Flash File System Generator, Revision 2.0
        Written by Eric Gregori

header size = 153
reading file: index.htm file size = 4263
header size = 151
reading file: pot_data.txt file size = 7
filename header length = 0x18
copying file: index_htm.htp to output
copying file: pot_data_txt.htp to output
D:\kirin2e_labs\runtime_loaded_web_page_example\ajax_graph_demo>pause
Press any key to continue . . .

D:\kirin2e_labs\runtime_loaded_web_page_example\ajax_graph_demo>emg_web_uploader
 192.168.1.99 dynamic.ffs joshua



Web page / Firmware Uploader for EMG HTTP server
   Written by Eric Gregori


Sending Erase Command
Sending Unlock Key

Waiting for Erase ACK
Erasing Flash 0x24000 block address
Erasing Flash 0x28000 block address
Erasing Flash 0x2C000 block address
Erasing Flash 0x30000 block address
Erasing Flash 0x34000 block address
Erasing Flash 0x38000 block address
Erasing Flash 0x3C000 block address
Erasing Flash 0x40000 block address
Erase Complete
Sending File dynamic.ffs to 192.168.1.99
4654 bytes sent

Waiting for Upload Complete
Upload Complete

4656 bytes successfully sent
Transmission Complete


D:\kirin2e_labs\runtime_loaded_web_page_example\ajax_graph_demo>pause
Press any key to continue . . .
```

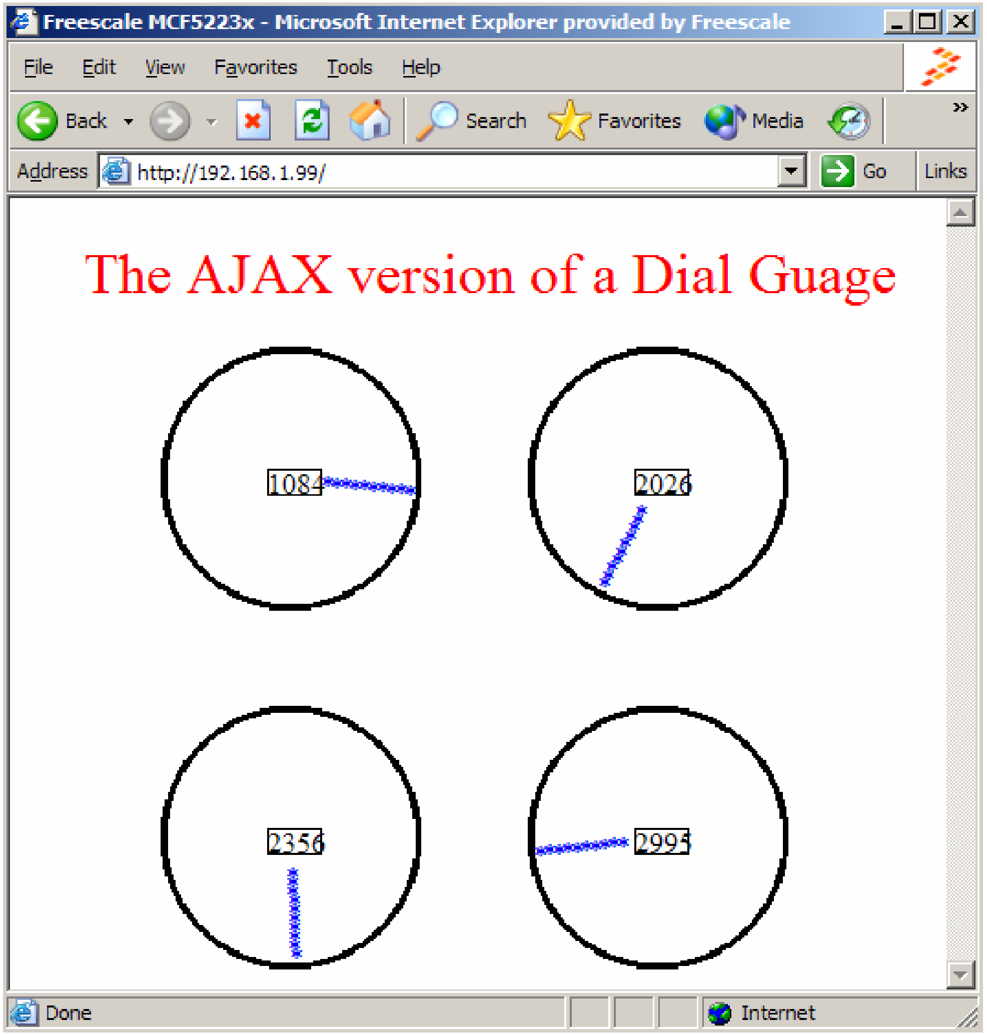
Turn the POT


Freescale MCF5223x - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.1.99/
Go Links
AJAX In Action on a MCF5223x
202.9
234.8
299.5
Done
Internet

LAB 12: Monitoring Analog Data with a dial guage

- Goto the LAB12_?????? Directory.
- Double Click the make.bat to load the LAB into the ColdFire.

Turn the POT, and move the board around


Freescale MCF5223x - Microsoft Internet Explorer provided by Freescale
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.1.99/ Go Links
The AJAX version of a Dial Guage
1084
2026
2356
2995
Done
Internet

LAB 13: Accessing files in the FFS

- Goto the LAB13_?????? Directory.
- Double Click the make.bat to load the LAB into the ColdFire.
- Go through the presentation

The Powerpoint presentation has been converted to HTML and Javascript. The presentation is being served up by the ColdFire.

FFS User API

The FFS has a User API for user applications to access the flash file system.

The FFS can be used to store any type of data, binary or ascii.

The user can store accel tables, nv parameters, configuration info, …

The information can be accessed by the firmware with a simple open call.

The user can update the information by doing a runtime file load.

Emg_open

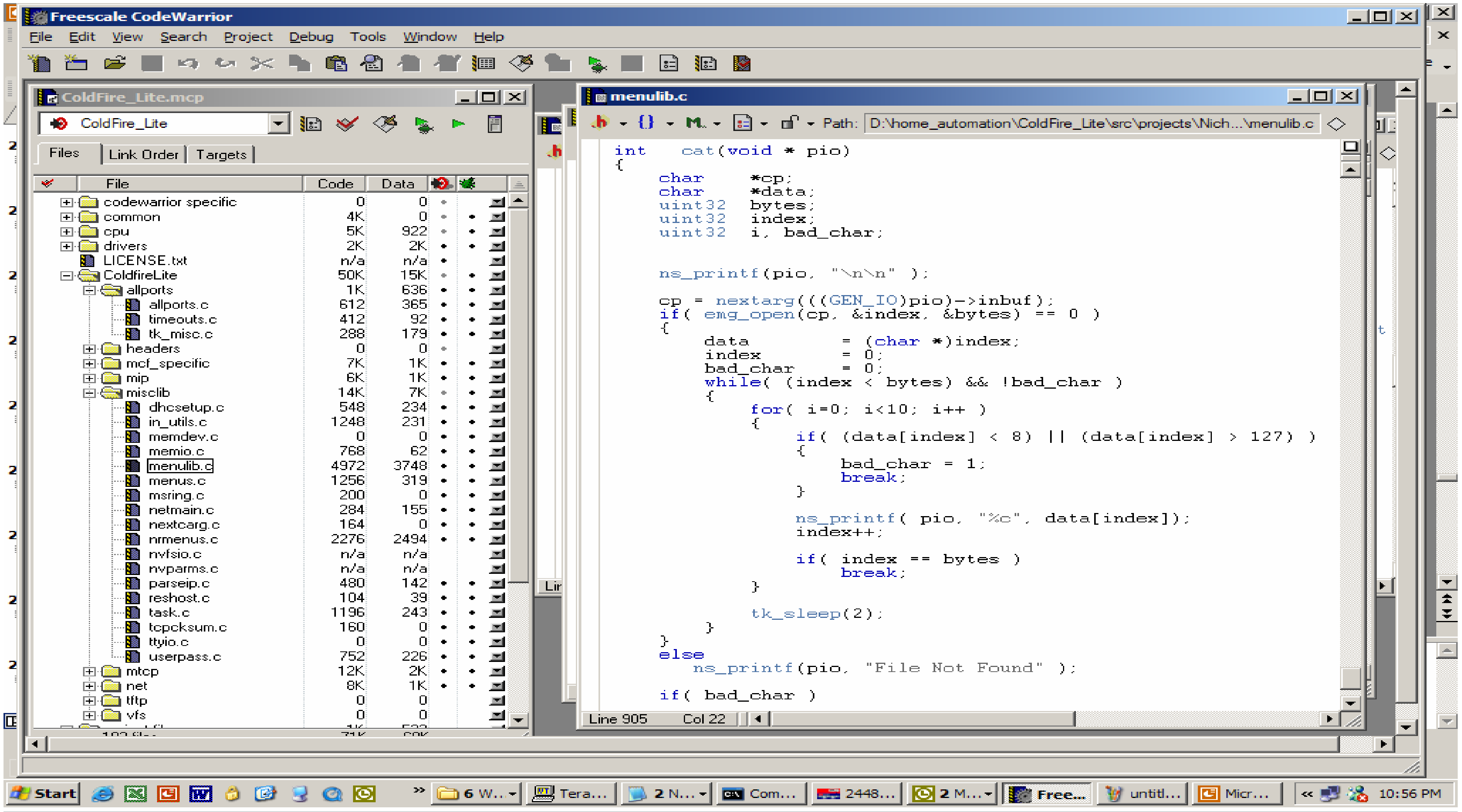
```
//******************************************************************************
// int emg_open( char *filename, uint32 *data_pointer, uint32 *file_size )
//
// User API to dynamic flash file system
//
// Finds the file descriptor in the FAT.
// Sets data_pointer to start of data.
// Sets file_size to size of file in bytes.
// returns a < 0 if error, 0 = success
//
// for an example of using emg_open(), see cat command in menulib.c
//
//
// Author: Eric Gregori  (847) 651 - 1971
//                       eric.gregori@freescale.com
//******************************************************************************
```

The CAT command

The CAT command is an example of how to use the emg_open() function.

The CAT command will dump the contents of a file to the console.

The CAT command code



Freescale CodeWarrior

ColdFire_Lite.mcp — ColdFire_Lite

Files | Link Order | Targets

File	Code	Data		
codewarrior specific	0	0	•	
common	4K	0	•	•
cpu	5K	922	•	•
drivers	2K	2K	•	•
LICENSE.txt	n/a	n/a	•	
ColdfireLite	50K	15K	•	•
allports	1K	636	•	•
allports.c	612	365	•	•
timeouts.c	412	92	•	•
tk_misc.c	288	179	•	•
headers	0	0	•	
mcf_specific	7K	1K	•	•
mip	6K	1K	•	•
misclib	14K	7K	•	•
dhcsetup.c	548	234	•	•
in_utils.c	1248	231	•	•
memdev.c	0	0	•	•
memio.c	768	62	•	•
menulib.c	4972	3748	•	•
menus.c	1256	319	•	•
msring.c	200	0	•	•
netmain.c	284	155	•	•
nextcarg.c	164	0	•	•
nrmenus.c	2276	2494	•	•
nvfsio.c	n/a	n/a		
nvparms.c	n/a	n/a		
parseip.c	480	142	•	•
reshost.c	104	39	•	•
task.c	1196	243	•	•
tcpcksum.c	160	0	•	•
ttyio.c	0	0	•	•
userpass.c	752	226	•	•
mtcp	12K	2K	•	•
net	8K	1K	•	•
tftp	0	0		
vfs	0	0		

menulib.c — Path: D:\home_automation\ColdFire_Lite\src\projects\Nich...\menulib.c

```c
int     cat(void * pio)
{
        char        *cp;
        char        *data;
        uint32      bytes;
        uint32      index;
        uint32      i, bad_char;


        ns_printf(pio, "\n\n" );

        cp = nextarg(((GEN_IO)pio)->inbuf);
        if( emg_open(cp, &index, &bytes) == 0 )
        {
                data            = (char *)index;
                index           = 0;
                bad_char        = 0;
                while( (index < bytes) && !bad_char )
                {
                        for( i=0; i<10; i++ )
                        {
                                if( (data[index] < 8) || (data[index] > 127) )
                                {
                                        bad_char = 1;
                                        break;
                                }

                                ns_printf( pio, "%c", data[index]);
                                index++;

                                if( index == bytes )
                                        break;
                        }

                        tk_sleep(2);
                }
        }
        else
           ns_printf(pio, "File Not Found" );

        if( bad_char )
```

Line 905   Col 22

LAB 14: Try to load a image > 128K

- Goto the LAB14_?????? Directory.
- Double Click the make.bat to load the LAB into the ColdFire.

The load will fail, because the image is too big.

Verify that the original dynamic FFS contents have not been corrupted.

FAQs

How many web pages can be loaded into the Run Time or Compile Time FFS?

- 255 files in each for a total of 510

What is the MAX size of a Run Time Web Page image?

- 128K, Limited only by the size of a flash logical block.

What is the MAX size of a Compile Time Web Page Image?

- Whatever FLASH is left over from the TCP/IP stack and Web Server Firmware minus the Run Time FFS area(128K) = Currently about 64K.

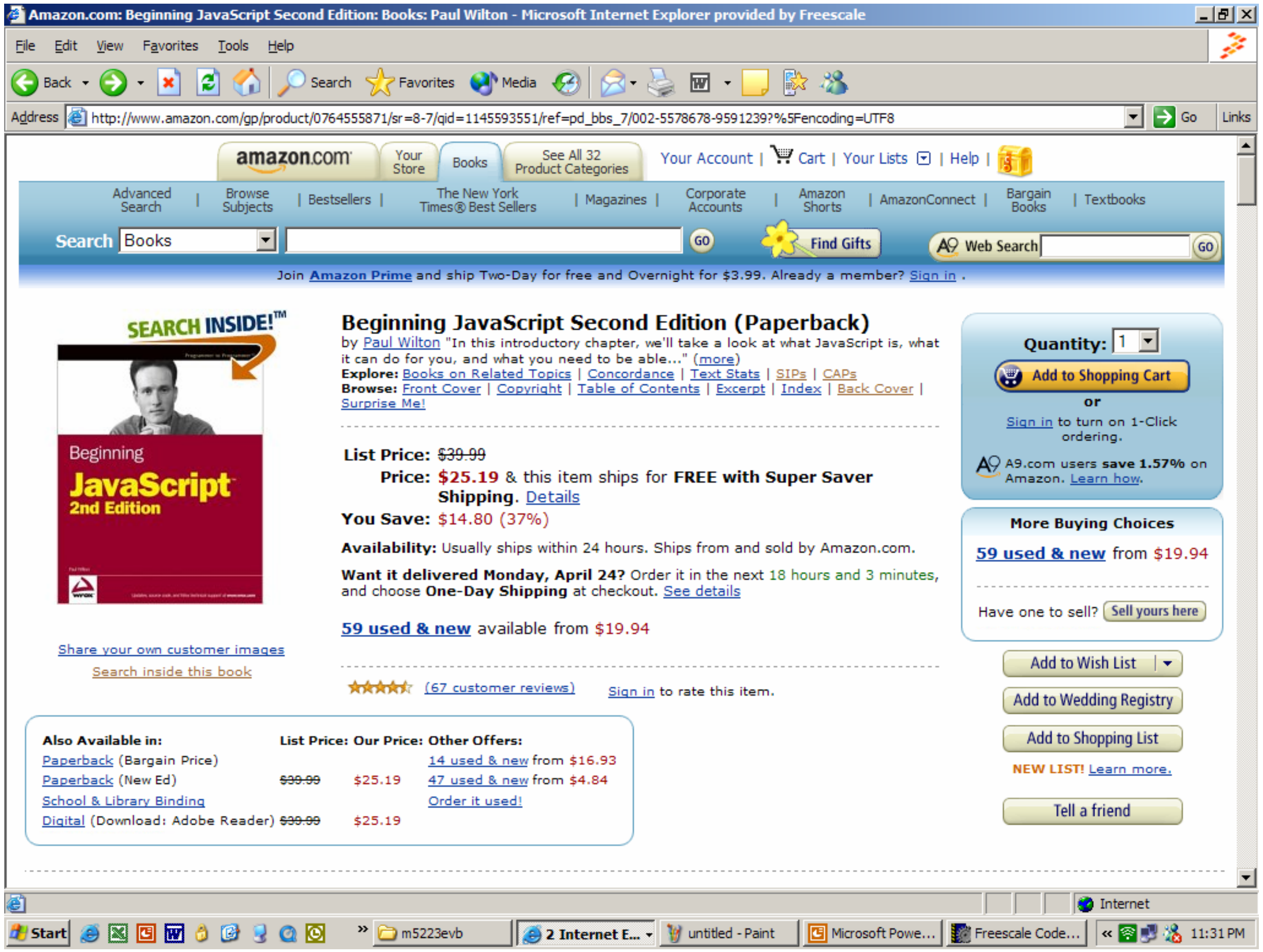
Is the Run Time Loadable Web Page verified after downloading?

- Yes and no. Handshaking is used to verify that all the pakets were transferred correctly. No, because there currently is no verify that flash got written correctly. There are hooks already in the code to do this, and I plan on releasing a update with these changes soon.

How quickly can AJAX poll the server for information?

- That depends on the connection, and the web browser. With a small closed network, and Internet Explorer 6.0, the update rate can be as high as 100ms.

Reference Material



Amazon.com: Beginning JavaScript Second Edition: Books: Paul Wilton - Microsoft Internet Explorer provided by Freescale

Address http://www.amazon.com/gp/product/0764555871/sr=8-7/qid=1145593551/ref=pd_bbs_7/002-5578678-9591239?%5Fencoding=UTF8

Join Amazon Prime and ship Two-Day for free and Overnight for $3.99. Already a member? Sign in .

Beginning JavaScript Second Edition (Paperback)
by Paul Wilton "In this introductory chapter, we'll take a look at what JavaScript is, what it can do for you, and what you need to be able..." (more)
Explore: Books on Related Topics | Concordance | Text Stats | SIPs | CAPs
Browse: Front Cover | Copyright | Table of Contents | Excerpt | Index | Back Cover | Surprise Me!

List Price: ~~$39.99~~
Price: $25.19 & this item ships for **FREE with Super Saver Shipping**. Details
You Save: $14.80 (37%)

Availability: Usually ships within 24 hours. Ships from and sold by Amazon.com.

Want it delivered Monday, April 24? Order it in the next 18 hours and 3 minutes, and choose **One-Day Shipping** at checkout. See details

59 used & new available from $19.94

(67 customer reviews) Sign in to rate this item.

Share your own customer images
Search inside this book

Also Available in:	List Price:	Our Price:	Other Offers:
Paperback (Bargain Price)			14 used & new from $16.93
Paperback (New Ed)	~~$39.99~~	$25.19	47 used & new from $4.84
School & Library Binding			Order it used!
Digital (Download: Adobe Reader)	~~$39.99~~	$25.19	

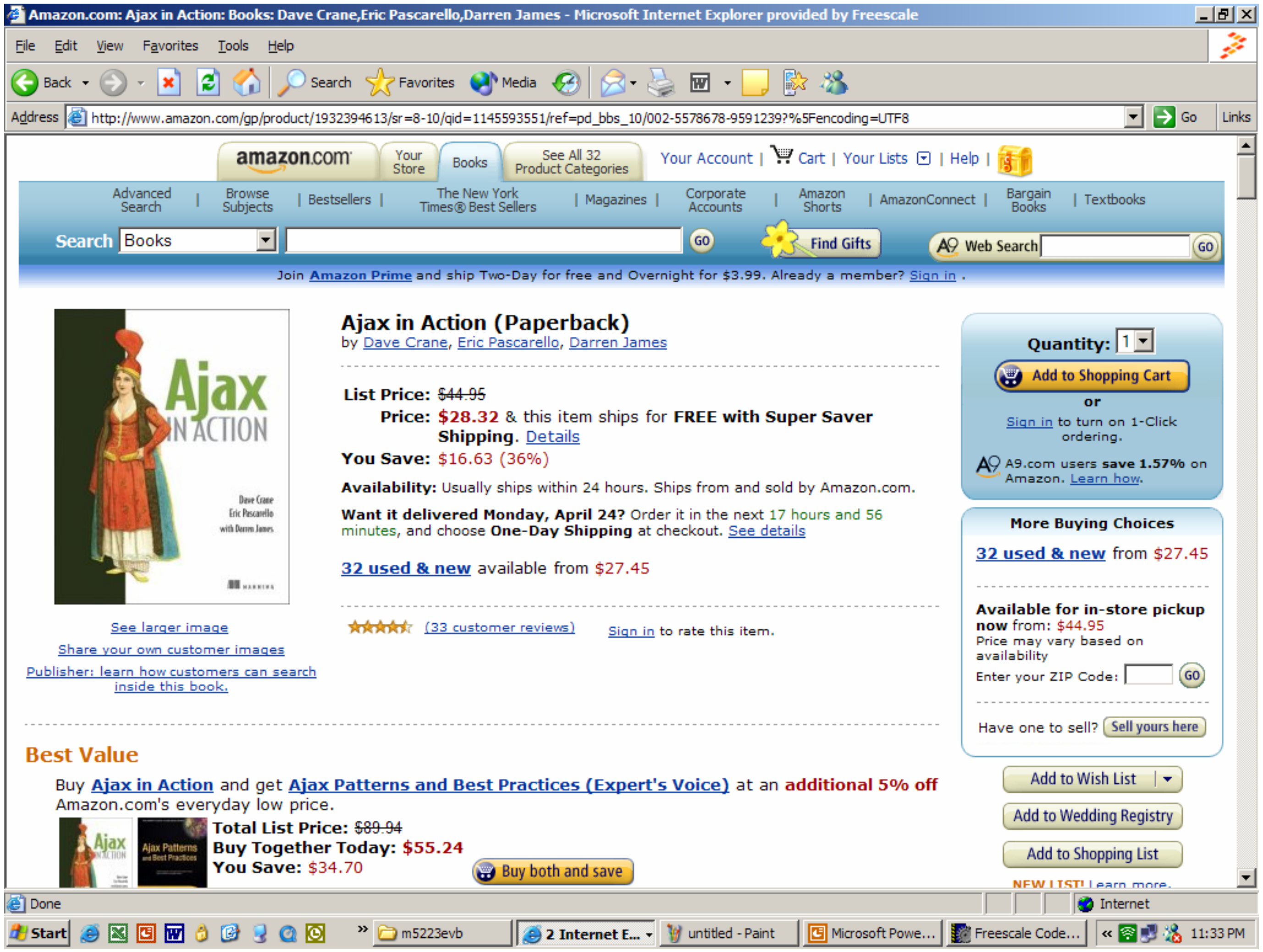
Quantity: 1
Add to Shopping Cart
or
Sign in to turn on 1-Click ordering.
A9.com users **save 1.57%** on Amazon. Learn how.

More Buying Choices
59 used & new from $19.94
Have one to sell? Sell yours here

Add to Wish List
Add to Wedding Registry
Add to Shopping List
NEW LIST! Learn more.
Tell a friend

File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
amazon.com Your Store Books See All 32 Product Categories Your Account | Cart | Your Lists | Help
Advanced Search | Browse Subjects | Bestsellers | The New York Times® Best Sellers | Magazines | Corporate Accounts | Amazon Shorts | AmazonConnect | Bargain Books | Textbooks
Search Books GO Find Gifts Web Search GO
SEARCH INSIDE!™
Beginning JavaScript 2nd Edition
Internet
Start m5223evb 2 Internet E... untitled - Paint Microsoft Powe... Freescale Code... 11:31 PM

Reference Material


Amazon.com: Ajax in Action: Books: Dave Crane,Eric Pascarello,Darren James - Microsoft Internet Explorer provided by Freescale

Address http://www.amazon.com/gp/product/1932394613/sr=8-10/qid=1145593551/ref=pd_bbs_10/002-5578678-9591239?%5Fencoding=UTF8

Join Amazon Prime and ship Two-Day for free and Overnight for $3.99. Already a member? Sign in .

Ajax in Action (Paperback)
by Dave Crane, Eric Pascarello, Darren James

List Price: ~~$44.95~~
Price: $28.32 & this item ships for **FREE with Super Saver Shipping**. Details
You Save: $16.63 (36%)

Availability: Usually ships within 24 hours. Ships from and sold by Amazon.com.

Want it delivered Monday, April 24? Order it in the next 17 hours and 56 minutes, and choose **One-Day Shipping** at checkout. See details

32 used & new available from $27.45

(33 customer reviews) Sign in to rate this item.

See larger image
Share your own customer images
Publisher: learn how customers can search inside this book.

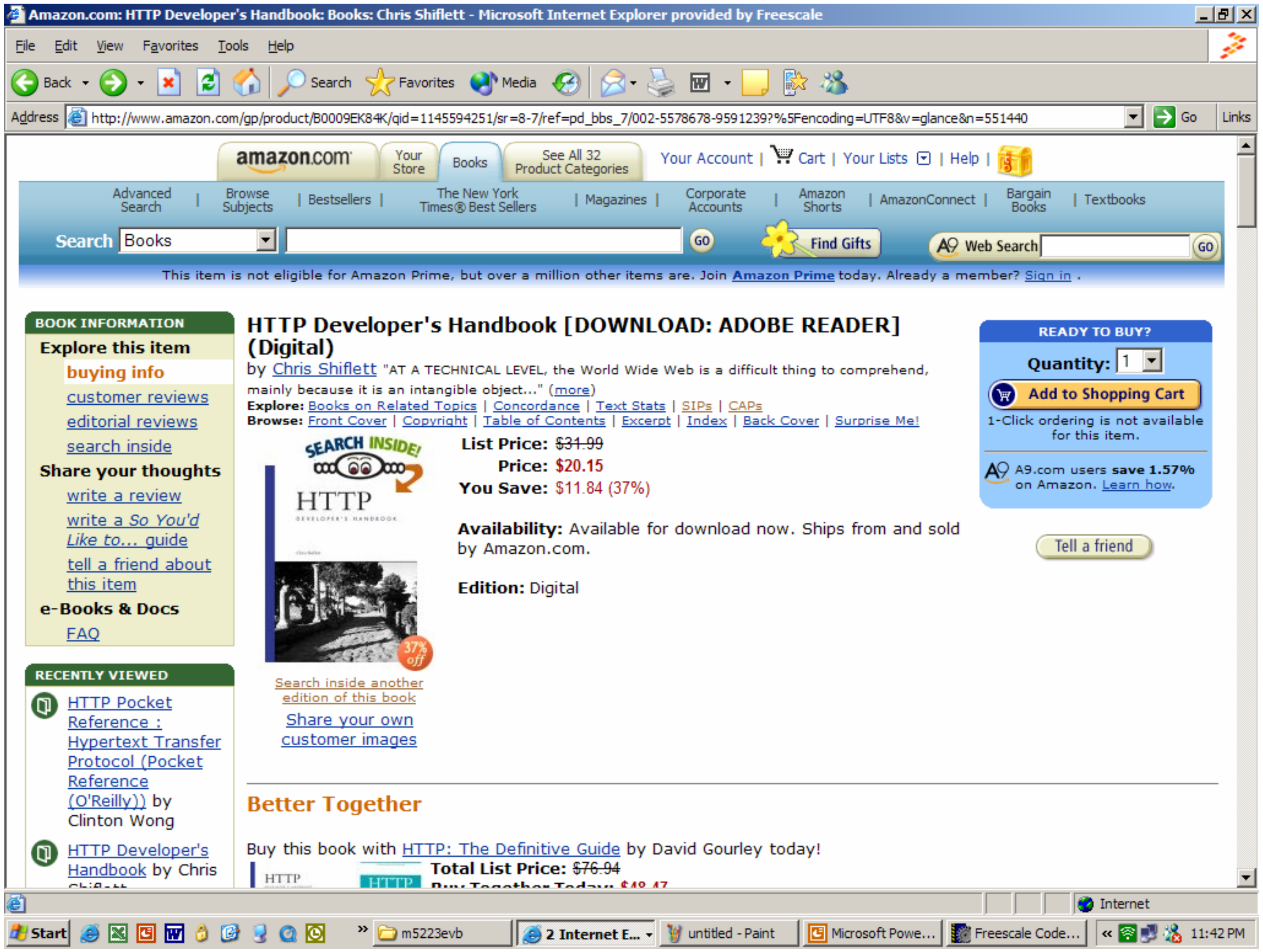
Quantity: 1
Add to Shopping Cart
or
Sign in to turn on 1-Click ordering.
A9.com users **save 1.57%** on Amazon. Learn how.

More Buying Choices
32 used & new from $27.45
Available for in-store pickup now from: $44.95
Price may vary based on availability
Enter your ZIP Code:
Have one to sell? Sell yours here

Best Value
Buy Ajax in Action and get Ajax Patterns and Best Practices (Expert's Voice) at an **additional 5% off** Amazon.com's everyday low price.
Total List Price: ~~$89.94~~
Buy Together Today: $55.24
You Save: $34.70
Buy both and save

File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
amazon.com Your Store Books See All 32 Product Categories
Your Account | Cart | Your Lists | Help
Advanced Search | Browse Subjects | Bestsellers | The New York Times® Best Sellers | Magazines | Corporate Accounts | Amazon Shorts | AmazonConnect | Bargain Books | Textbooks
Search Books GO Find Gifts Web Search GO
Add to Wish List
Add to Wedding Registry
Add to Shopping List
Done Internet
Start m5223evb 2 Internet E... untitled - Paint Microsoft Powe... Freescale Code... 11:33 PM

Reference Material



Amazon.com: HTTP Developer's Handbook: Books: Chris Shiflett - Microsoft Internet Explorer provided by Freescale

Address: http://www.amazon.com/gp/product/B0009EK84K/qid=1145594251/sr=8-7/ref=pd_bbs_7/002-5578678-9591239?%5Fencoding=UTF8&v=glance&n=551440

This item is not eligible for Amazon Prime, but over a million other items are. Join Amazon Prime today. Already a member? Sign in .

BOOK INFORMATION

Explore this item
- buying info
- customer reviews
- editorial reviews
- search inside

Share your thoughts
- write a review
- write a So You'd Like to... guide
- tell a friend about this item

e-Books & Docs
- FAQ

RECENTLY VIEWED
- HTTP Pocket Reference : Hypertext Transfer Protocol (Pocket Reference (O'Reilly)) by Clinton Wong
- HTTP Developer's Handbook by Chris

HTTP Developer's Handbook [DOWNLOAD: ADOBE READER] (Digital)

by Chris Shiflett "AT A TECHNICAL LEVEL, the World Wide Web is a difficult thing to comprehend, mainly because it is an intangible object..." (more)

Explore: Books on Related Topics | Concordance | Text Stats | SIPs | CAPs

Browse: Front Cover | Copyright | Table of Contents | Excerpt | Index | Back Cover | Surprise Me!

List Price: ~~$31.99~~

Price: $20.15

You Save: $11.84 (37%)

Availability: Available for download now. Ships from and sold by Amazon.com.

Edition: Digital

Search inside another edition of this book

Share your own customer images

READY TO BUY?

Quantity: 1

Add to Shopping Cart

1-Click ordering is not available for this item.

A9.com users **save 1.57%** on Amazon. Learn how.

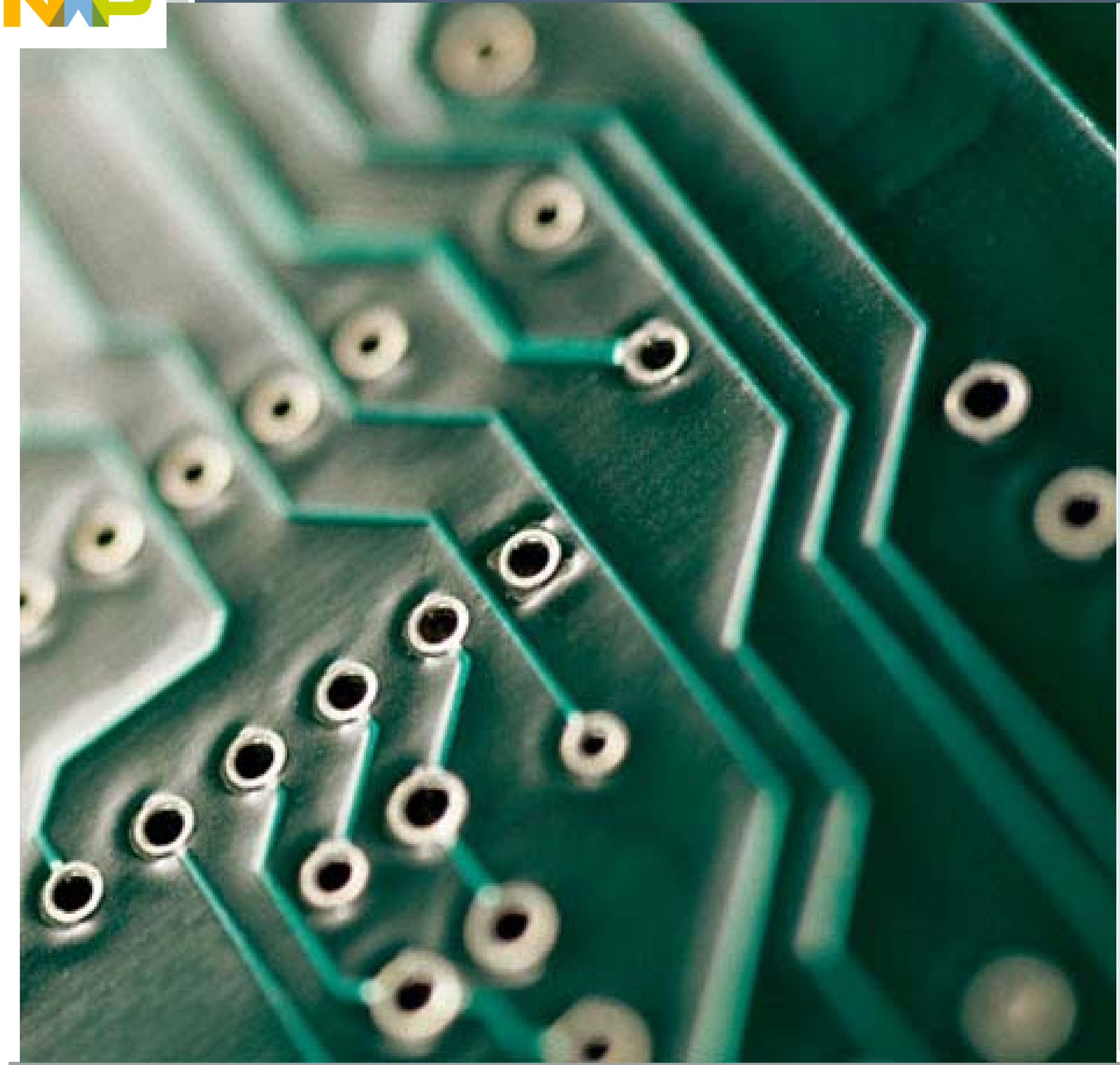
Tell a friend

Better Together

Buy this book with HTTP: The Definitive Guide by David Gourley today!

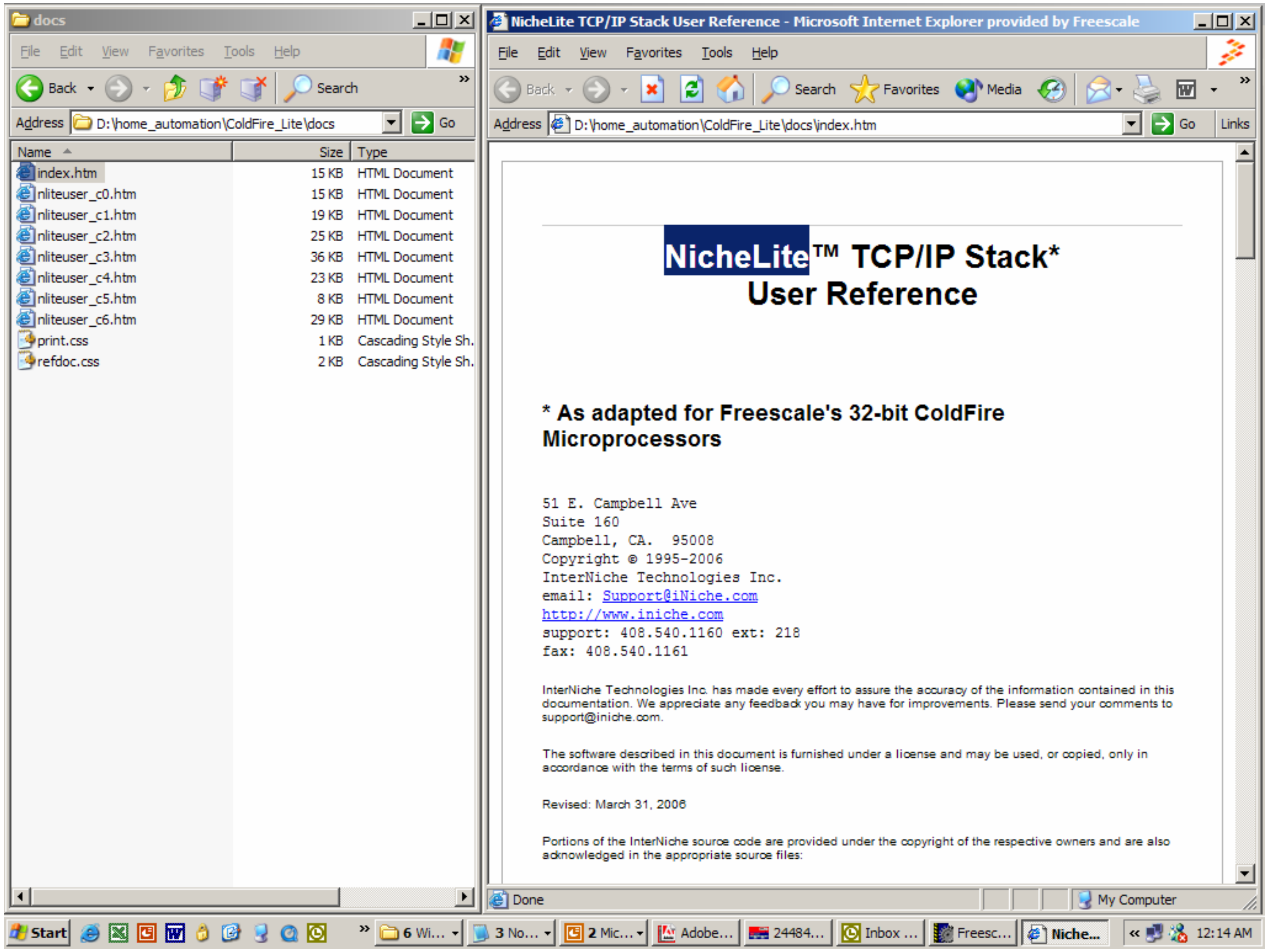
Total List Price: ~~$76.94~~

File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Go
Links
Your Store
Books
See All 32 Product Categories
Your Account | Cart | Your Lists | Help
Advanced Search | Browse Subjects | Bestsellers | The New York Times® Best Sellers | Magazines | Corporate Accounts | Amazon Shorts | AmazonConnect | Bargain Books | Textbooks
Search Books
GO
Find Gifts
Web Search
SEARCH INSIDE!
HTTP
37% off
Internet
Start
m5223evb
2 Internet E...
untitled - Paint
Microsoft Powe...
Freescale Code...
11:42 PM



Firmware Overview

NicheLite Documentation can be found in the project



Address: D:\home_automation\ColdFire_Lite\docs

Name	Size	Type
index.htm	15 KB	HTML Document
nliteuser_c0.htm	15 KB	HTML Document
nliteuser_c1.htm	19 KB	HTML Document
nliteuser_c2.htm	25 KB	HTML Document
nliteuser_c3.htm	36 KB	HTML Document
nliteuser_c4.htm	23 KB	HTML Document
nliteuser_c5.htm	8 KB	HTML Document
nliteuser_c6.htm	29 KB	HTML Document
print.css	1 KB	Cascading Style Sh.
refdoc.css	2 KB	Cascading Style Sh.

NicheLite TCP/IP Stack User Reference - Microsoft Internet Explorer provided by Freescale

Address: D:\home_automation\ColdFire_Lite\docs\index.htm

NicheLite™ TCP/IP Stack*
User Reference

* As adapted for Freescale's 32-bit ColdFire Microprocessors

```
51 E. Campbell Ave
Suite 160
Campbell, CA.  95008
Copyright © 1995-2006
InterNiche Technologies Inc.
email: Support@iNiche.com
http://www.iniche.com
support: 408.540.1160 ext: 218
fax: 408.540.1161
```

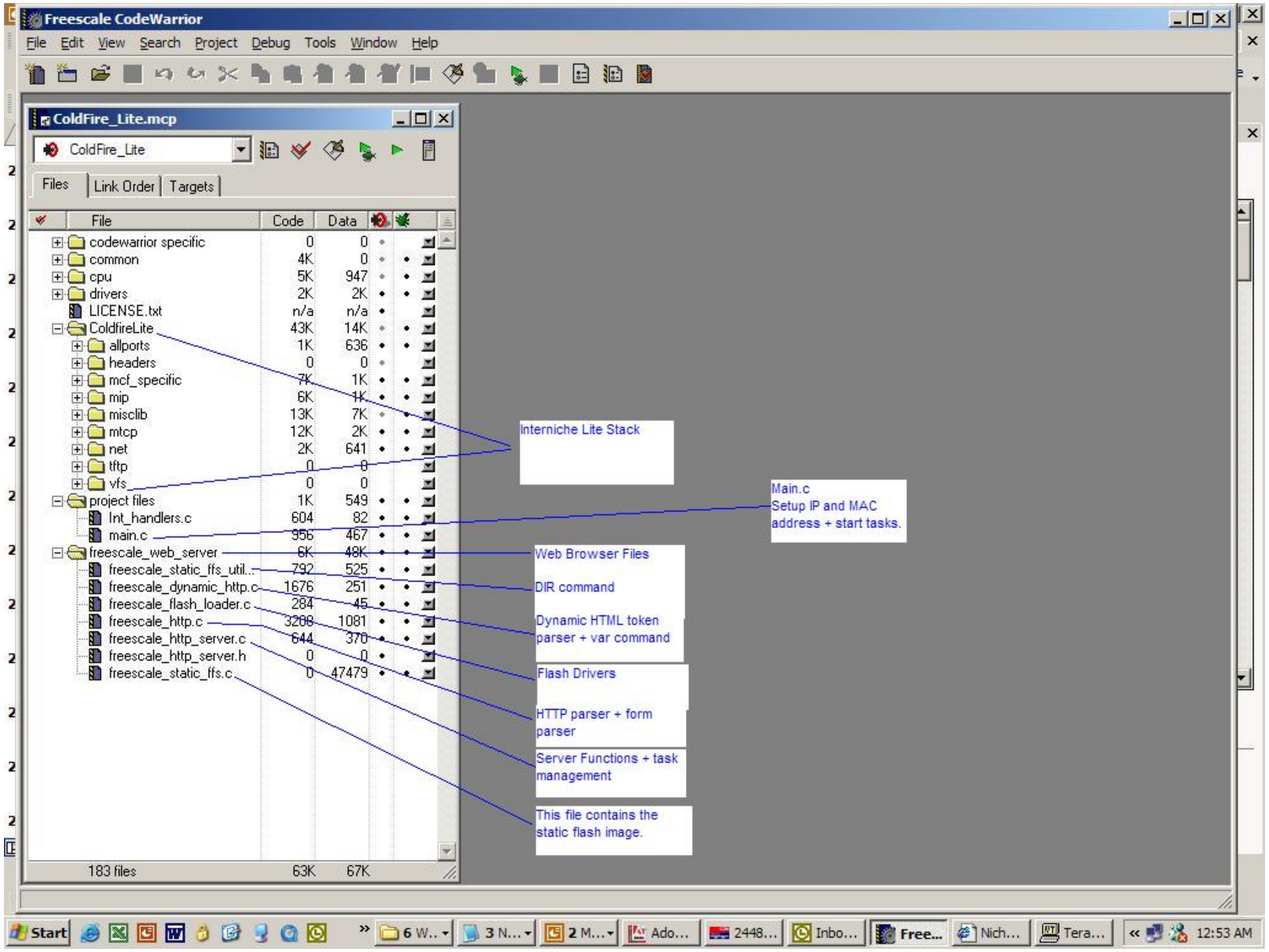
InterNiche Technologies Inc. has made every effort to assure the accuracy of the information contained in this documentation. We appreciate any feedback you may have for improvements. Please send your comments to support@iniche.com.

The software described in this document is furnished under a license and may be used, or copied, only in accordance with the terms of such license.

Revised: March 31, 2006

Portions of the InterNiche source code are provided under the copyright of the respective owners and are also acknowledged in the appropriate source files:

Project Files



File	Code	Data		
codewarrior specific	0	0	•	
common	4K	0	•	•
cpu	5K	947	•	•
drivers	2K	2K	•	•
LICENSE.txt	n/a	n/a	•	
ColdfireLite	43K	14K	•	•
allports	1K	636	•	•
headers	0	0	•	
mcf_specific	7K	1K	•	•
mip	6K	1K	•	•
misclib	13K	7K	•	•
mtcp	12K	2K	•	•
net	2K	641	•	•
tftp	0	0		
vfs	0	0		
project files	1K	549	•	•
Int_handlers.c	604	82	•	•
main.c	956	467	•	•
freescale_web_server	6K	48K	•	•
freescale_static_ffs_util...	792	525	•	•
freescale_dynamic_http.c	1676	251	•	•
freescale_flash_loader.c	284	45	•	•
freescale_http.c	3208	1081	•	•
freescale_http_server.c	644	370	•	•
freescale_http_server.h	0	0	•	
freescale_static_ffs.c	0	47479	•	•
183 files	63K	67K		

Interniche Lite Stack

Main.c
Setup IP and MAC address + start tasks.

Web Browser Files

DIR command

Dynamic HTML token parser + var command

Flash Drivers

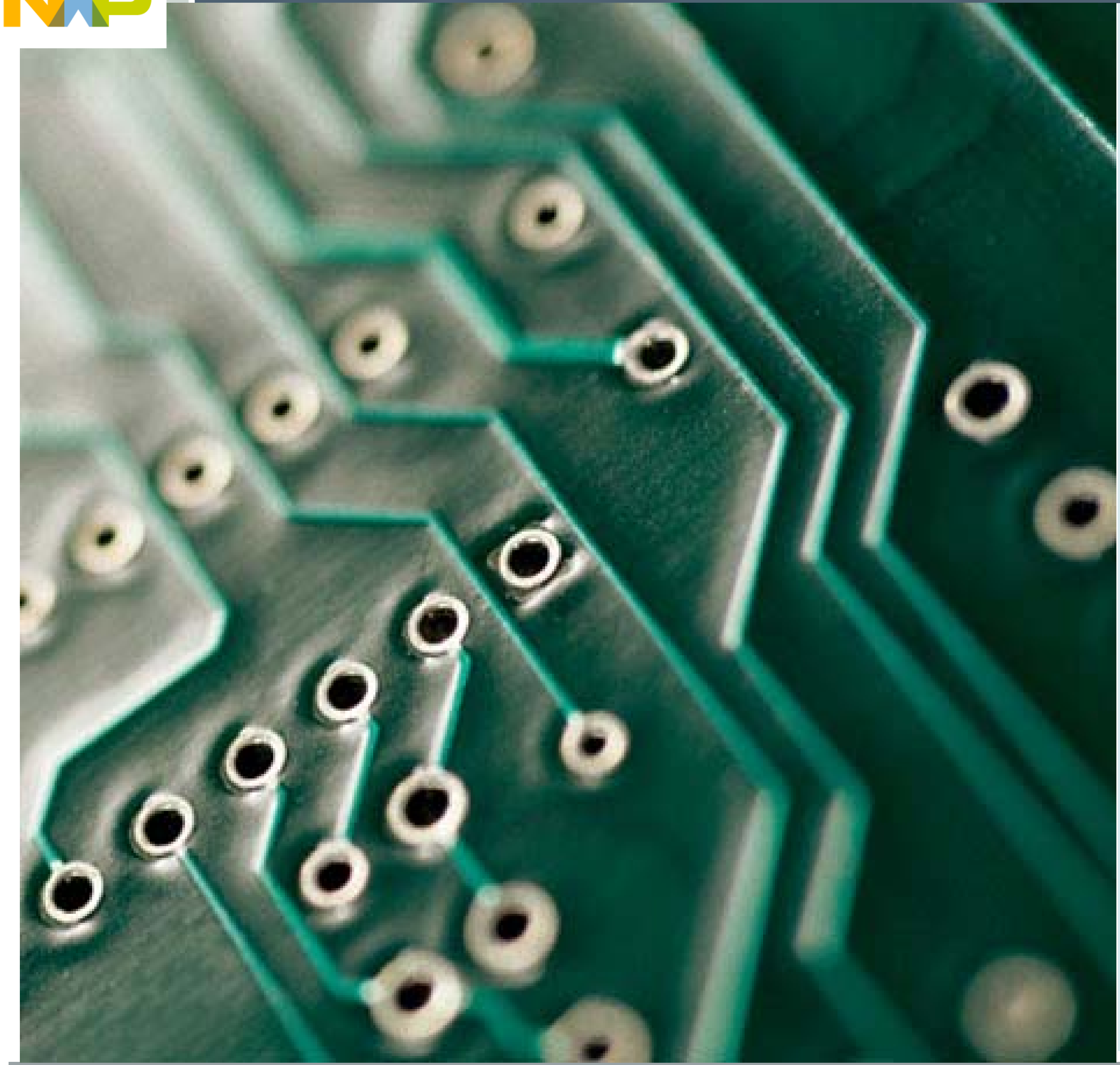
HTTP parser + form parser

Server Functions + task management

This file contains the static flash image.

The HTTP Server Task

```
//*****************************************************************************
// Declare Task Object
//*****************************************************************************
TK_OBJECT(to_emghttpsrv);
TK_ENTRY(tk_emghttpsrv);
struct inet_taskinfo emg_http_task = {
                                        &to_emghttpsrv,
                                        "EMG HTTP server",
                                        tk_emghttpsrv,
                                        NET_PRIORITY,
                                        APP_STACK_SIZE
                                        };
long emghttpsrv_wakes = 0;


TK_ENTRY(tk_emghttpsrv)
{
   int err;

   while (!iniche_net_ready)
      TK_SLEEP(1);

   err = freescale_http_init();
   if( err == SUCCESS )
   {
      exit_hook(freescale_http_cleanup);
   }
   else
   {
      dtrap();                                                    // emghttp_init()  shouldn't ever fail
   }

   for (;;)
   {
      freescale_http_check();                                     // will block on select
      tk_yield();                                                 // give up CPU in case it didn't block

     emghttpsrv_wakes++;                                                         //

      if (net_system_exit)
         break;
   }
   TK_RETURN_OK();
}
```



Questions, Answers and Consultations

freescale™
semiconductor