

# 基于 MCF52259 的以太网和 USB 应用

HTTP/TFTP 服务器 + USB 主机大容量存储

作者: Li Kan

32 位应用程序部门

## 1 引言

该应用报告描述了基于 ColdFire MCF5225x 处理器的 HTTP/TFTP 服务器的实施方案；该实施方案还集成了用以保存网页或其他文件的 USB 文件系统。本报告详细介绍如何将 USB 堆栈与 NicheTask 集成起来，以及如何通过 HTTP/TFTP 服务器访问 USB 文件系统。

## 2 MCF52259

MCF52259 是基于 V2 ColdFire 微架构的高度集成的 32 位微控制器，包括 64 KB 的内部 SRAM，512 KB 的 Flash 内存，1 个快速以太网控制器，1 个 USB OTG 控制器，1 个外部总线接口，4 个 32 位定时器，1 个 4 通道 DMA 控制器，2 个 IIC 模块，3 个 UART 和 1 个队列 SPI。MCF52259 系列适用于通用的工业控制应用。

## 3 堆栈

本应用程序采用快速以太网控制器和 USB OTG 控制器，并在堆栈上获得 InterNiche 和 CMX 提供的所有支持。

### 目录

1	引言	1
2	MCF52259	1
3	堆栈	1
3.1	InterNiche	2
3.2	CMX USB-Lite	2
4	HTTP 服务器实施方案	2
4.1	项目体系架构	2
4.2	实施方案	2
4.3	在 MCF52259EVB 上运行 HTTP Web 服务器	7
4.4	连接到 USB Flash 的网页上	9
5	TFTP 服务器	9
5.1	软件体系架构	9
5.2	实施方案	10
5.3	项目编译	13
5.4	在 MCF52259EVB 上运行 TFTP 服务器	13
5.5	将文件上载到 TFTP 服务器上	14
5.6	从 TFTP 服务器上下载文件	14
6	结论	15

## 3.1 InterNiche

ColdFire® TCP/IP 堆栈是公共源堆栈，可应用于 ColdFire 系列处理器。它可以分成两大部分：小型 TCP 层库和小型 IP 层库。另外，它还包括一个虚拟文件系统（VFS），该系统支持美国信息互换标准码（ASCII）和二进位数据，并且与名为 NicheTask 的轮询任务系统集成。如需了解 ColdFire TCP/IP 堆栈的更多信息，请参见应用笔记“AN3470 oldFire TCP/UDP/IP Stack and RTOS and AN3455 oldFire Lite HTTP Server”。

## 3.2 CMX USB-Lite

CMX USB-Lite 由飞思卡尔提供，CMX 可应用于 ColdFire 和 S08 系列的 USB 微控制器。它分别支持 USB 器件、主机和 OTG 功能，以满足不同的设计需求。它还内带键盘鼠标等通用人机接口设备（HID）和 USB 通用异步收发器（UART）的通信设备类（CDC）设备的高级驱动，以及主机模式下的大容量存储产品的演示。如需获取该堆栈的更多信息，请访问：

[http://www.freescale.com/webapp/sps/site/prod\\_summary.jsp?code=CMX\\_USB-LITE&fsrch=1](http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CMX_USB-LITE&fsrch=1)。

# 4 HTTP 服务器实施方案

## 4.1 项目体系架构

本项目要通过 CodeWarrior 7.0 创建。项目架构详见表 1。

表 1. HTTP Web 服务器项目体系架构

模块	描述
CodeWarrior 规范	M52259EVB 的链接器文件。
ColdFirelite	InterNiche 堆栈，内含多个文件夹，包括 Ethernet TCP/IP 堆栈、支持的协议及 RTOS 处理器独立的文件。
通用模块	标准 C 功能库和 UART 输入和输出支持。
CPU	包含单独的处理器相关文件。
驱动	MII 接口的相关驱动
Freescall_Web_Server	HTTP Web 服务器的应用程序。
项目文件	主程序和中断源文件。
CMXUSB_LITE	包含多个文件夹的 CMX USB 堆栈，提供 USB 主机大容量存储的支持。

## 4.2 实施方案

该项目基于 MCF52259EVB 进行实施。

### 4.2.1 InterNiche 和 CMX USB 堆栈集成

在本项目中，InterNiche 被用来管理网络协议，CMX 堆栈被用来管理基于 USB 文件系统的文件操作。项目要点是将两个堆栈组合起来，并同时操作。

### 4.2.1.1 为 NicheTask 创建 usb-mass-host-task.c

InterNiche 还提供名为 NicheTask 的 RTOS，这是一款协作型多任务处理调度程序。在本体系架构中，每个任务有自己的任务堆栈。它能自动处理控制并返回给主调度程序。集成的方法是通过 NicheTask 调度将来自 CMX 堆栈的 API 任务。这是通过一个名为 usb-mass-host-task.c 的源文件实现的，其中还为 USB 主机大容量存储创建了状态机。该文件为 NicheTask 提供多个 API，具体功能详见表 2。

表 2. 适用于 InterNiche 的 CMX API

功能	描述
void demo_process(void)	适用于 USB 主机大容量存储的状态机
int cmd_dir(void * pio)	struct menu_op* usbmassmenu 命令
int cmd_dump(void * pio)	
int cmd_type(void * pio)	
TK_ENTRY(tk_cmxmasshosttask)	nicheTask 定义的任务对象，使用来自 CMX-USB-LITE 和 InterNiche 的 API。
void create_cmxusb_task( void )	在 NicheTask TCB 表中添加 CMX 任务，并且每次循环执行一次。

### 4.2.1.2 从 NicheTask 的 usb-mass-host-task.c 中调用 API

在 usb-mass-host-task.c 中，创建一个名为 create\_cmxusb\_task() 的功能，它被用来支持 NicheTask。它是从支持 NicheTask 的标准输入 / 输出任务对象 tk\_keyboard 调用，上电复位后它就被添加到任务控制模板 (TCB) 表格里。

```
TK_ENTRY(tk_keyboard)
{
    for (;;)
    {
        TK_SLEEP(1);           /* make keyboard yield some time */
        kbdiio();             /* let Iniche menu routines poll for char */
        keyboard_wakes++;     /* count wakeups */
#ifdef MCF52259
        if((usbmst_attach == 1)&&(usbtisk_created == 0))
        {
            create_cmxusb_task();
            usbtisk_created = 1;
        }
        else if((usbmst_attach == 0)&&(usbtisk_created == 1))
        {
            tk_kill(to_cmxmasshosttask);
            usbtisk_created = 0;
        }
#endif
        if (net_system_exit)
            break;
    }
    TK_RETURN_OK();
}
```

### 4.2.1.3 为 CMX 和 OS 集成添加定时器 tick 的定义

NicheTask tk\_sleep(tick) 被用来延迟任务，在 CMX host\_ms\_delay(ms) 中也几乎用于相同目的。但它们延迟的时间不相同。一个 tick 等于 5 ms。为了实现集成，host\_ms\_delay(ms) 将被 tk\_sleep(tick) 取代，而且要在 usb\_host.c 中进行修改。

```
void host_ms_delay(hcc_u32 delay)
{
#ifdef MCF52259
    delay/=5;          // 5 ms is used for 1 tick
    delay+=1;         //there might be a decimal fraction cut as 0 after the division, add 1
    tk_sleep(delay);
#else
    start_mS_timer((hcc_u16)delay);
    do {
    } while(!check_mS_timer());
#endif
}
```

### 4.2.2 添加 USB 文件类型的文件操作

Freescal Web\_Server 模块文件操作只支持 html 和文本文件。在本项目中，网页可以保存到 USB 器件里。因此，freescal\_http\_server.h 会增加一个新的文件类型定义，该定义支持名为 FILE\_TYPE\_DYANMIC3 的 USB 文件类型。

```
// File types scanned for dynamic HTML content
#define FILE_TYPE_DYANMIC1          'html'
#define FILE_TYPE_DYANMIC2          'text'
#ifdef MCF52259
#define FILE_TYPE_DYANMIC3          'usb'
#endif
```

功能函数 emg\_web\_open() 被用来打开网页。在打开的网页代码中加入新的部分，以支持定义为 FILE\_TYPE\_DYANMIC3 的新文件类型。为避免原来的代码结构遭到破坏，插入的代码要附加一个定义为 MCF52259\_MST 的宏。

```
void emg_web_open( unsigned char session, unsigned char *filename )
{
...
#ifdef MCF52259_MST
if(usbmst_attach == 1)
{
    F_FIND find;
    int ret;
    volatile unsigned char* fname;
    F_FILE *file;

    if(filename[0] == 0)
        fname = (volatile unsigned char*)"index.htm";
    else
        fname = filename;
#ifdef HTTP_VERBOSE>4
    printf("open session %d, %s\n\r",session,fname);
#endif
    file=f_open((const char*)fname, "r");
    if(file!=0)
    {
        freescal_http_sessions[session].file_type = FILE_TYPE_DYANMIC3;
    }
}
#endif
}
```

```

    freescale_http_sessions[session].file_size = f_filelength((const char*)fname);
    freescale_http_sessions[session].file_pointer = (void *)file; //used here is a file pointer
to pass the file handle
    freescale_http_sessions[session].file_index = 0;
    freescale_http_sessions[session].state = EMG_HTTP_STATE_SEND_FILE;
    found = 1;
    return;
}
} //if(usbmst_attach == 1)
#endif
...
}

```

功能函数 `emg_web_read()` 支持 html/ 文本文件读取操作。创建名为 `usb_web_read()` 的功能函数，在 USB 文件上完成数据读取操作。这些应用函数位于 `freescale_file_api.c` 中。

```

unsigned long usb_web_read( unsigned char session, unsigned char *buffer_out, unsigned long
max_bytestoread )
{
    unsigned long    i, k, j;
    unsigned long    read_index, bytes_to_read, last_read, read_size;
    unsigned char    *data;
    unsigned long r;

    if(freescale_http_sessions[session].file_index>=freescale_http_sessions[session].file_size )
    {
        return(0);
    }

    if( (freescale_http_sessions[session].file_index + max_bytestoread) >
        freescale_http_sessions[session].file_size )
    {
        max_bytestoread = freescale_http_sessions[session].file_size -
freescale_http_sessions[session].file_index;
    }

    r=f_read(buffer_out,1,max_bytestoread,(F_FILE*)(freescale_http_sessions[session].file_pointer
));
    /*#if HTTP_VERBOSE>4
    printf("session %d read %d\n\r",session,r);
    #endif
    freescale_http_sessions[session].file_index+=r;
    return r;
}
#endif

```

### 4.2.3 在 CMX 中添加 USB 检测器件

HTTP 网络服务器还能在内部 Flash 上保存网页。如果 USB 器件没连接上，该应用程序也可以从内部读取网页。应用程序在 USB 中断伺服程序中添加名为 `usbmst_attach` 的变量，可用来表示 USB 器件是否连接上，通过该变量可以向该应用程序通告 USB 器件是否连接上。

```

extern unsigned char usbmst_attach;
__declspec(interrupt)
void usb_it_handler(void) //
{

```

```
unsigned char istr;
/* Save irq USB status. */
//istr=MCF_USB_INT_STAT;
if(MCF_USB_INT_STAT & MCF_USB_INT_STAT_ATTACH)
{
    MCF_USB_INT_ENB &= ~MCF_USB_INT_ENB_ATTACH;//clear interrupt flag
    usbmst_attach = 1;          //tell applications that a usb flash is attached
}
}
```

因为在复位后，USB 器件连接中断被禁用，所以必须在 `main()` 初始化时重新启用。为了实现这个目标，需要配置两个寄存器：其中一个是 `MCF_USB_INT_STAT`，另一个是 `MCF_USB_INT_ENB`。

```
int main(void)
{
...
MCF_USB_INT_STAT = MCF_USB_INT_STAT_ATTACH; //write to clear it
MCF_USB_INT_ENB |= MCF_USB_INT_ENB_ATTACH;
...
}
```

### 4.2.4 在网页上显示传感器数据和网络状态

因为本项目在 `MCF52259EVb` 上实施，而且在该板卡上，可变电阻器同 `ADC` 端口相连，所以可以用来收集数据，并提供给与之关联的 `Web` 服务器。而且该服务器还能在网页上提供网络统计数据。用来收集数据的代码位于名为 `evb_specific_collect_sensor_data()` 的功能函数内。该函数可以在 `MCF52259_evb.c` 中找到。`ADC` 收集数据由名为 `read_AD()` 的功能函数实现。与网络状态相关的信息则来自数据结构 `ip_mib`，该结构被 `InterNiche` 堆栈用来保存简单网络管理协议（`SNMP`）的 `IP` 状态。

```
void evb_specific_collect_sensor_data(void) //FSL new function for evb specific implementation
{
...
    html_vars[3] = read_AD(0);
    html_vars_flags[3] = 1;

    html_vars[4] = read_AD(1);
    html_vars_flags[4] = 1;

    html_vars[5] = read_AD(2);
    html_vars_flags[5] = 1;

    html_vars[6] = read_AD(3);
    html_vars_flags[6] = 1;

    html_vars[7] = read_AD(4);
    html_vars_flags[7] = 1;

    html_vars[8] = read_AD(5);
    html_vars_flags[8] = 1;

    html_vars[9] = read_AD(6);
    html_vars_flags[9] = 1;

    html_vars[10] = read_AD(7);
    html_vars_flags[10] = 1;

...
}
```

```

html_vars[14] = ip_mib.ipInReceives;          /* total received datagrams (bad too) */
html_vars_flags[14] = 1;

html_vars[15] = ip_mib.ipInHdrErrors;        /* Header Err (xsum, ver, ttl, etc) */
html_vars_flags[15] = 1;

html_vars[16] = ip_mib.ipInAddrErrors;      /* nonsense IP addresses */
html_vars_flags[16] = 1;

html_vars[17] = ip_mib.ipUnknownProtos;     /* unknown protocol types */
html_vars_flags[17] = 1;

html_vars[18] = ip_mib.ipInDelivers;        /* delivered receive packets */
html_vars_flags[18] = 1;

html_vars[19] = ip_mib.ipOutRequests;       /* sends (not including routed) */
html_vars_flags[19] = 1;

html_vars[20] = ip_mib.ipOutDiscards;       /* sends dropped (no buffer) */
html_vars_flags[20] = 1;

html_vars[21] = ip_mib.ipOutNoRoutes;       /* dropped, does not route */
html_vars_flags[21] = 1;
}

```

## 4.2.5 项目编译

源代码中插入“4.2.1、4.2.2 和 4.2.3 小节”所列的修改。为避免原代码结构遭到破坏，它们全部要附加一个名为 MCF52259 的宏。在编译项目前，请先对这个宏进行预定义。

```
#define MCF52259
```

## 4.3 在 MCF52259EVB 上运行 HTTP Web 服务器

在运行 HTTP Web 服务器之前，必须通过 UAR T0 端口将开发评估板（EVB）连到电脑的串口上。超级终端的波特率必须设为 115200-8N1。在上电复位后，将显示相关信息，详见图 1。

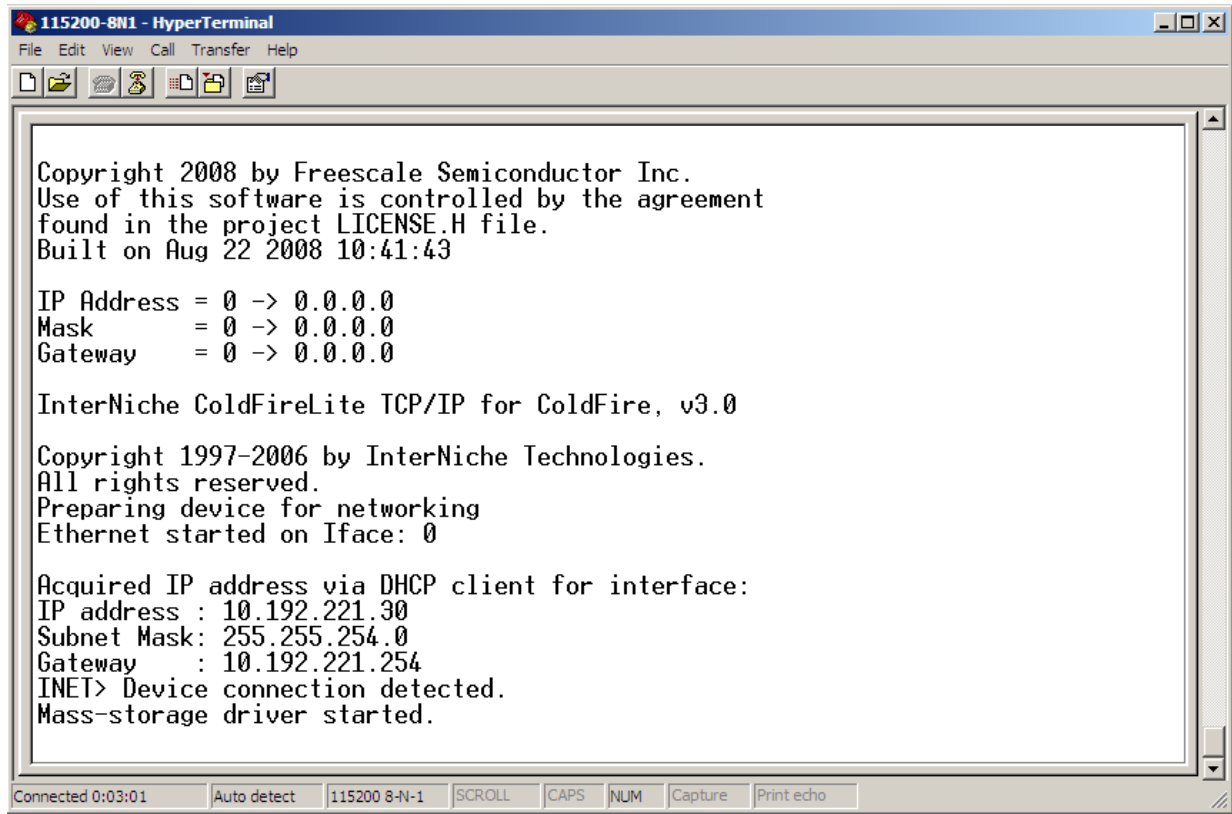


图 1. 超级终端

HTTP Web 服务器还包括 DHCP 客户端，因此如果网络上有 DHCP 服务器的话，它就能自动获取一个 IP 地址。在图 1 中，该服务器的 IP 地址是 10.192.221.30。如果 USB 器件已经插在 EVB 上，这时还会输出与此有关的信息。

## 4.4 连接到 USB Flash 的网页上

由于在上一步中已得到 HTTP Web 服务器的 IP 地址，所以直接在地址栏中输入 <http://10.192.221.30>，就能通过 IE 访问该网址。

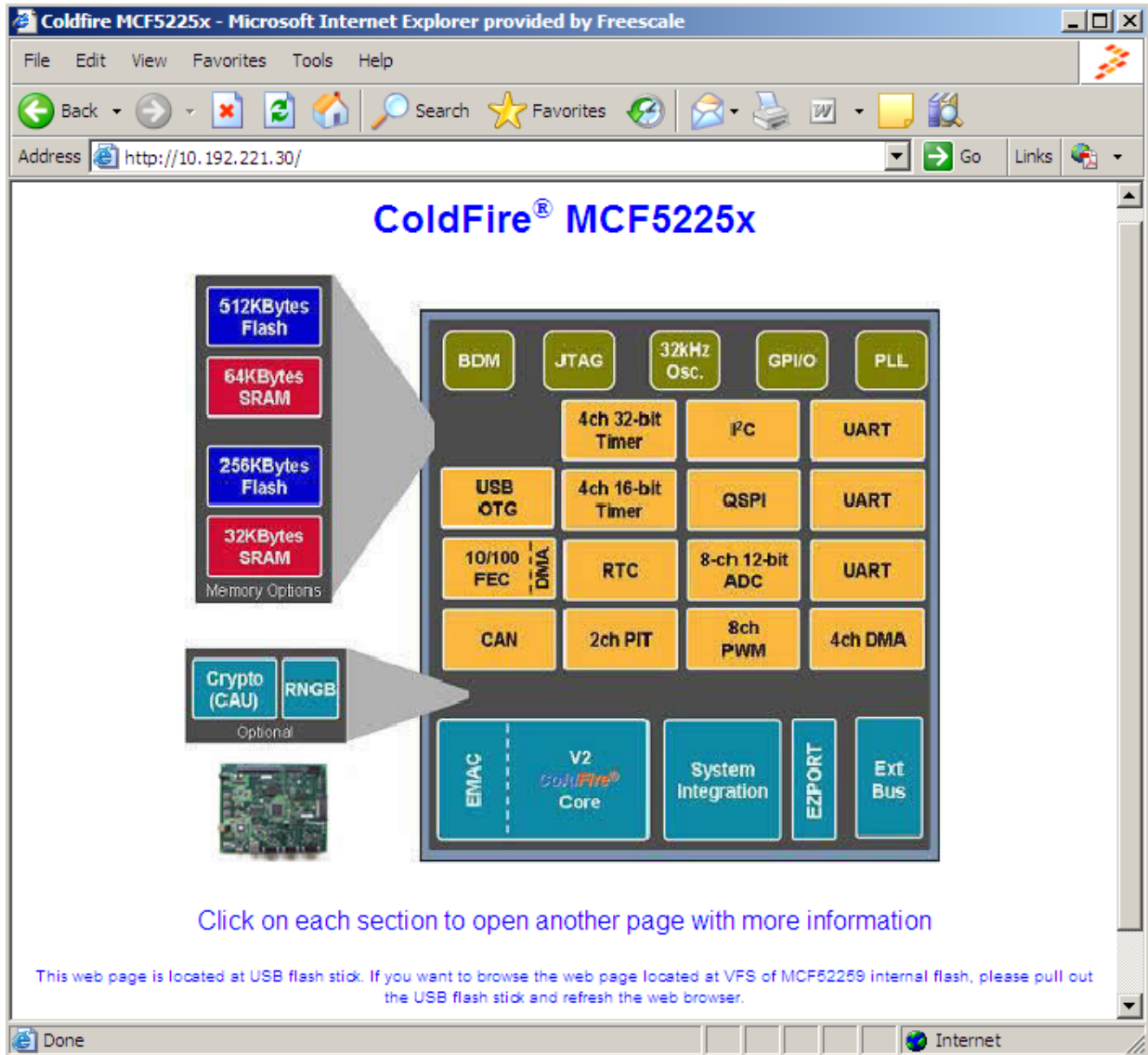


图 2. 互联网网页

## 5 TFTP 服务器

### 5.1 软件体系架构

该架构与 HTTP Web 服务器相似。主要区别在于它支持 TFTP 协议（而不是 HTTP 协议），具体如表 3 所示。

表 3. TFTP 服务器项目体系架构

模块	描述
CodeWarrior 规范	M52259EVB 的链接器文件
ColdFirelite	InterNiche 堆栈, 内含多个文件夹, 包括 Ethernet TCP/IP 堆栈、支持的协议及 RTOS 处理器独立的文件
通用模块	标准 C 功能库和 UART 输入 / 输出支持
CPU	包含单独的处理器相关文件
驱动	MII 接口的驱动
项目文件	主程序和中断源文件
CMXUSB_LITE	CMX USB 堆栈, 包括多个文件夹, 提供 USB 主机大容量存储的支持

## 5.2 实施方案

本项目也是在 MCF52259EVB 的基础上实施的。InterNiche 已经提供了 TFTP 服务器。原样例是通过 VFS 访问保存在内部 Flash 上的文件, 因而对文件大小有严格限制。在本项目中, 文件保存到 USB 器件上, 因此可保存更大型的文件。其中实现的要点是让 TFTP 服务器通过 CMX 文件系统 (而不是 VFS) 访问 USB 文件。

### 5.2.1 为 USB 文件添加文件操作支持

在本项目中, 与文件操作相关的 API 已被 CMX 文件系统取代。

#### 5.2.1.1 用 CMX 文件系统取代 VFS 里的文件描述符

文件描述符是文件系统使用的重要参数。VFILE 是为 VFS 定义的, 现在已经被 CMX 文件系统取代了。

```
#ifndef MCF52259_MST
    F_FILE*tf_fd;
#else
    VFILE *    tf_fd;                /* file descriptor for xfer */
#endif
```

### 5.2.1.2 用 CMX 文件系统取代 VFS 里的文件打开功能

功能 `vfopen()` 在 VFS 中提供的文件打开功能，现在由 CMX 文件系统的 `f_open()` 提供。

```
#ifndef MCF52259_MST
#include "thin_usr.h"
#endif
...
#ifdef MCF52259_MST //
    cn->tf_fd = f_open(fname, "w");
    #else
    cn->tf_fd = vfopen(fname, "w");
    #endif
else if(mode == OCTET)
    #ifdef MCF52259_MST
    cn->tf_fd = f_open(fname, "w");
    #else
    cn->tf_fd = vfopen(fname, "wb");
    #endif
...
if(mode == ASCII)
    #ifdef MCF52259_MST
    cn->tf_fd = f_open(fname, "r");
    #else
    cn->tf_fd = vfopen(fname, "r");
    #endif
else if(mode == OCTET)
    #ifdef MCF52259_MST
    cn->tf_fd = f_open(fname, "r");
    #else
    cn->tf_fd = vfopen(fname, "rb");
    #endif
else
    return("Invalid mode");
...

```

### 5.2.1.3 用 CMX 文件系统取代 VFS 里的文件关闭功能

功能函数 `vfclose()` 在 VFS 中提供的文件关闭功能，现在由 CMX 文件系统里的 `f_close()` 提供。

```
if(cn->tf_fd != NULL)
#ifdef MCF52259_MST
    f_close(cn->tf_fd);
    #else
    vfclose(cn->tf_fd);
#endif

```

### 5.2.1.4 用 CMX 文件取代 VFS 里的文件写入功能

功能函数 `vfwrite()` 提供的 VFS 文件读写功能，现在由 CMX 文件系统的 `f_write()` 提供。

```
#ifdef MCF52259_MST
    if((int)f_write(data, 1, len, cn->tf_fd) != (int)len)
    #else
    if((int)vfwrite(data, 1, len, cn->tf_fd) != (int)len)
#endif
```

### 5.2.1.5 用 CMX 文件取代 VFS 里的文件读取功能

功能函数 `vfread()` 提供的 VFS 文件读取功能，现在由 CMX 文件系统的 `f_read()` 提供。

```
/* load file data into tftp buffer */
if(!cn->tf_NR) /* if this is NOT a retry, read in new data */
{
    #ifdef MCF52259_MST
    bytes = f_read(tfdata->tf_data, 1, NORMLEN, cn->tf_fd);
    #else
    bytes = fread(tfdata->tf_data, 1, NORMLEN, cn->tf_fd); /* read next block from file */
    #endif
    if(bytes < NORMLEN) /* end of file? */
    {
        #ifndef MCF52259_MST
        if(vferror(cn->tf_fd)) /* check if it is an error */
        {
            if(cn->callback)
                cn->callback(TFC_FILEREAD, cn, "file read error");
            return FALSE;
        }
        #endif
    }
    #endif /* else at End Of File; fall through to do the last send */

    cn->tf_flen = bytes; /* bytes in last packet sent */
    cn->tf_size += bytes; /* total bytes sent so far */
}
...
```

## 5.2.2 在 NicheTask 中添加 TFTP 服务器任务

预定义 `TFTP_SERVER` 来激活 TFTP 服务器代码。不要忘记取消与 VFS 相关的宏定义。

```
#ifdef TFTP_PROJECT
#define TFTP_SERVER 1 /* include TFTP server code */
//#define VFS_FILES 1 /* include Virtual File System */
#endif
```

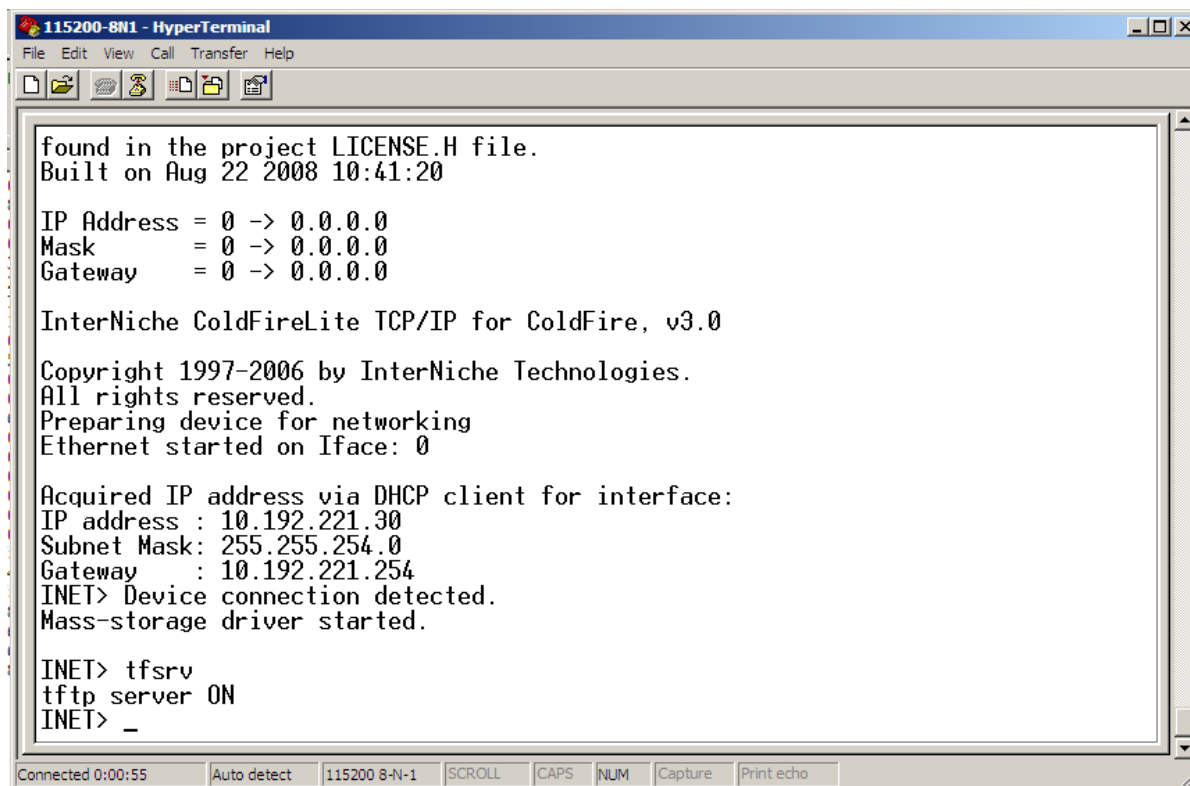
## 5.3 项目编译

同 HTTP 项目类似，在编写项目之前必须完成预定义。

```
#define TFTP_PROJECT  
#define MCF52259_MST
```

## 5.4 在 MCF52259EVB 上运行 TFTP 服务器

在运行 TFTP 服务器之前，必须通过 UART0 端口将 EVB 连到电脑串口上。超级终端的波特率必须设为 115200-8N1。在上电复位后，出现的信息将如 图 3 所示。



```
115200-8N1 - HyperTerminal  
File Edit View Call Transfer Help  
found in the project LICENSE.H file.  
Built on Aug 22 2008 10:41:20  
  
IP Address = 0 -> 0.0.0.0  
Mask = 0 -> 0.0.0.0  
Gateway = 0 -> 0.0.0.0  
  
InterNiche ColdFireLite TCP/IP for ColdFire, v3.0  
  
Copyright 1997-2006 by InterNiche Technologies.  
All rights reserved.  
Preparing device for networking  
Ethernet started on Iface: 0  
  
Acquired IP address via DHCP client for interface:  
IP address : 10.192.221.30  
Subnet Mask: 255.255.254.0  
Gateway : 10.192.221.254  
INET> Device connection detected.  
Mass-storage driver started.  
  
INET> tfsrv  
tftp server ON  
INET> _  
  
Connected 0:00:55 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

图 3. 超级终端

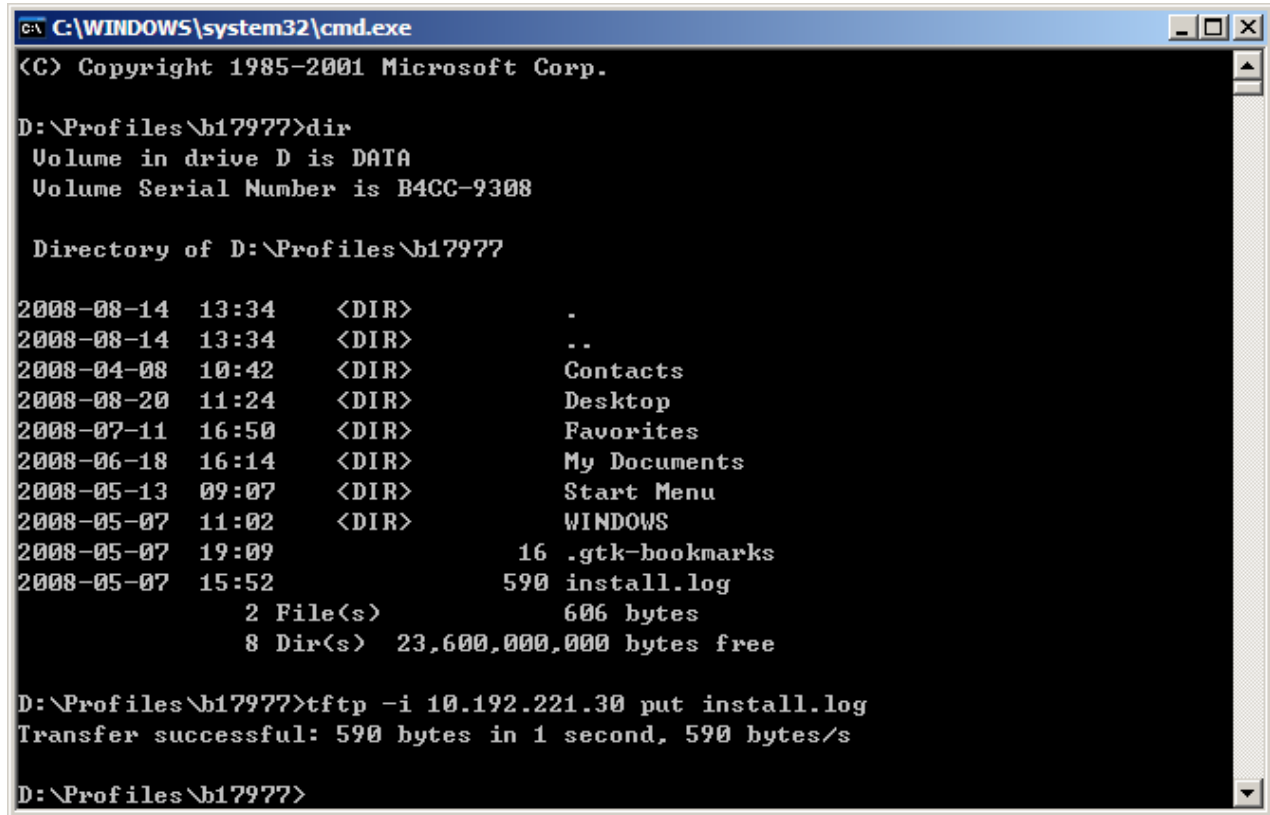
TFTP 服务器还包括了 DHCP 客户端。它能自动获取 IP 地址。在 图 3 中，该服务器的 IP 地址是 10.192.221.30。如果 USB 器件已经连接到 EVB 上，那么就会输出相关信息。然后可输入命令“tfsrv”来激活 TFTP 服务器。

## 5.5 将文件上传到 TFTP 服务器上

在获取 TFTP 服务器的 IP 地址后，可以通过 DOS 命令 `tftp -i <dst_addr> put filename.xx` 将文件上传到服务器上。在图 3 所示的样例中，`dst_addr` 地址是 10.192.221.30，文件名可以是已打开文件夹的任何一个文件。

### 注意

千万不要忘记上面的 DOS 命令选项 `-i`!



```
C:\WINDOWS\system32\cmd.exe
(C) Copyright 1985-2001 Microsoft Corp.

D:\Profiles\b17977>dir
Volume in drive D is DATA
Volume Serial Number is B4CC-9308

Directory of D:\Profiles\b17977

2008-08-14 13:34 <DIR> .
2008-08-14 13:34 <DIR> ..
2008-04-08 10:42 <DIR> Contacts
2008-08-20 11:24 <DIR> Desktop
2008-07-11 16:50 <DIR> Favorites
2008-06-18 16:14 <DIR> My Documents
2008-05-13 09:07 <DIR> Start Menu
2008-05-07 11:02 <DIR> WINDOWS
2008-05-07 19:09          16 .gtk-bookmarks
2008-05-07 15:52          590 install.log
                2 File(s)          606 bytes
                8 Dir(s) 23,600,000,000 bytes free

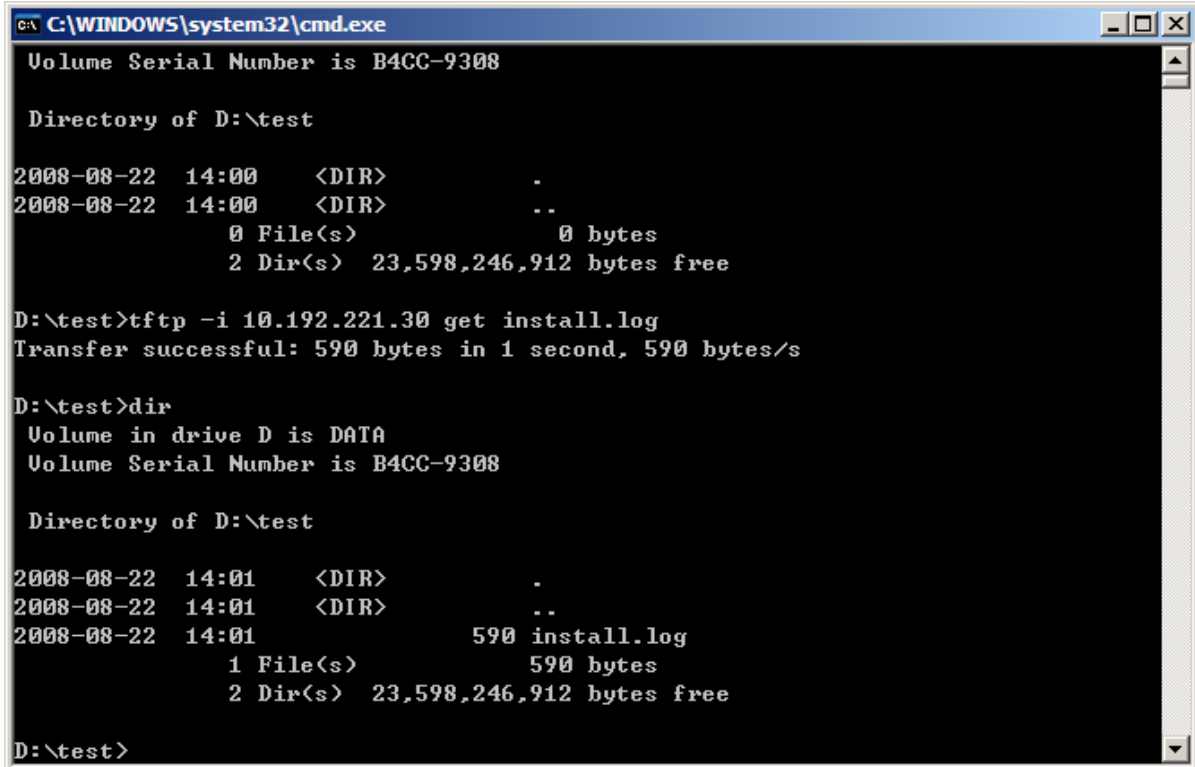
D:\Profiles\b17977>tftp -i 10.192.221.30 put install.log
Transfer successful: 590 bytes in 1 second, 590 bytes/s

D:\Profiles\b17977>
```

图 4. 将文件上传到 TFTP 服务器上

## 5.6 从 TFTP 服务器上下载文件

与上传文件类似，使用 DOS 命令 `tftp -i <dst_addr> get filename.xx`，可以从服务器下载文件。



```
C:\WINDOWS\system32\cmd.exe
Volume Serial Number is B4CC-9308

Directory of D:\test

2008-08-22  14:00    <DIR>        .
2008-08-22  14:00    <DIR>        ..
                0 File(s)        0 bytes
                2 Dir(s)  23,598,246,912 bytes free

D:\test>tftp -i 10.192.221.30 get install.log
Transfer successful: 590 bytes in 1 second, 590 bytes/s

D:\test>dir
Volume in drive D is DATA
Volume Serial Number is B4CC-9308

Directory of D:\test

2008-08-22  14:01    <DIR>        .
2008-08-22  14:01    <DIR>        ..
2008-08-22  14:01                590 install.log
                1 File(s)        590 bytes
                2 Dir(s)  23,598,246,912 bytes free

D:\test>
```

图 5. 从 TFTP 服务器上下载文件

## 6 结论

HTTP/TFTP 服务器项目是在 MCF52259 的 FEC 和 USB 模块基础上实现的，它们不但性能高而且提供灵活的连接。基于 MCF52259 的应用不仅限于 HTTP/TFTP 服务器。如果将连接 USB Flash 改为连接 USB 摄像机到 MCU 上，那么只需要对 HTTP/TFTP 服务器做最小的修改，就可以将 Web 监控器设置好。它可以通过网页监控正在发生的事情。MCF52259 是工业控制应用的理想之选。根据本应用笔记，还可以完成更多与网络应用相关的开发。

## 联系我们：

### 主页：

[www.freescale.com](http://www.freescale.com)

### 技术支持网站：

<http://www.freescale.com/support>

### 美国 / 欧洲或未列出的地点：

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### 欧洲、中东和非洲：

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### 日本：

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### 亚太地区：

飞思卡尔半导体（中国）有限公司  
北京市朝阳区建国路乙 118 号京汇大厦 23 层 100022  
+86 10 5879 8000  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### 索取技术资料：

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or +1-303-675-2140  
Fax: +1-303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

文档号：AN3779ZHS  
Rev. 0  
1/2009

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

