AN204414

LwIP over Ethernet on FM Family

Associated Part Family:	Series	Product Number
	MB9BF210	MB9BF216/17/18
	MB9BF610	MB9BF616/17/18
	MB9BFD10	MB9BFD16/17/18
	S6E2CC	
	S6E2C2	
	S6E2G	

Ethernet hardware needs a software protocol stack to be used for exchanging data. This application note describes the operation of the popular free open-source TCP/IP stack LwIP (Lightweight IP), version 1.4.1 on Cypress FM Family Microcontrollers.

Contents

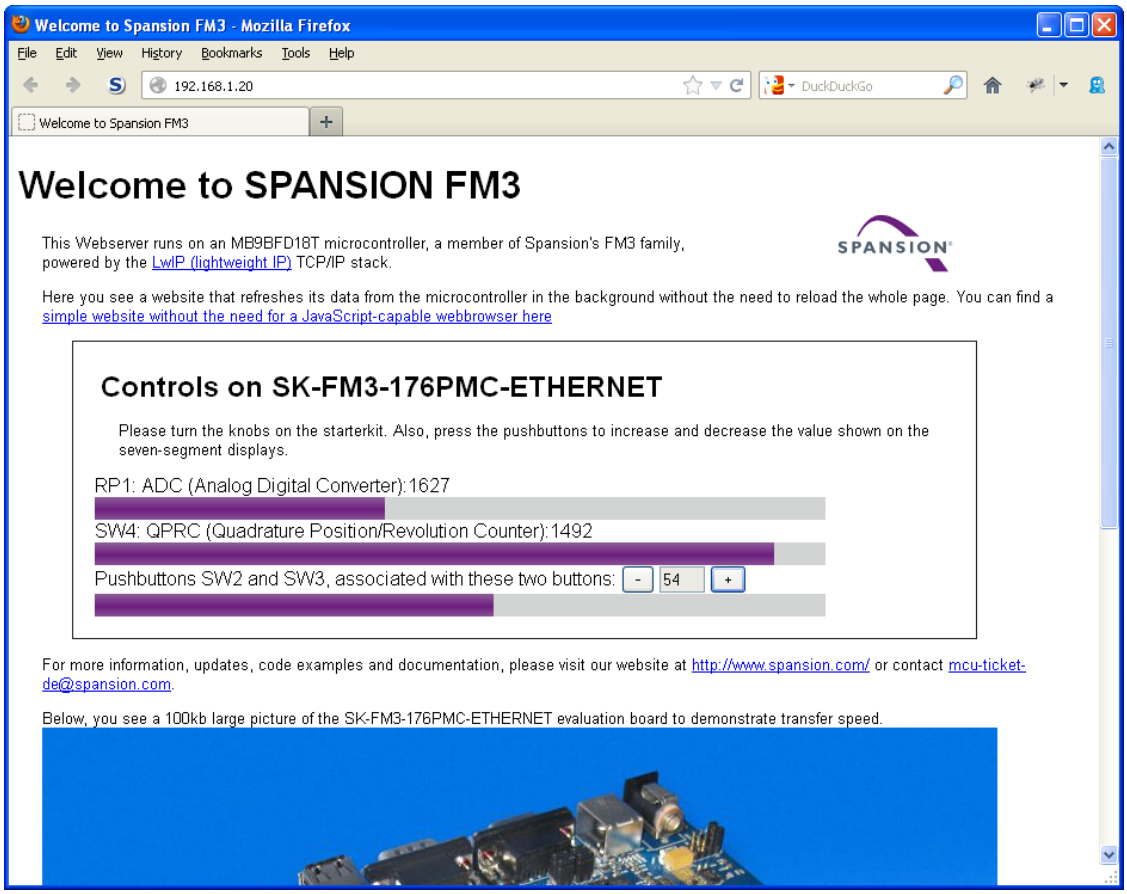
1 Introduction

Some types of the Cypress FM microcontroller family feature up to two independent controllers for IEEE802.3 Ethernet. This application note describes some important aspects to know for using this hardware solution together with the free-licensed open-source TCP/IP stack LwIP (lightweight IP) 1.4.0.

For a comprehensive description of the hardware and a programming guide, please consult the microcontroller's hardware manual, Ethernet section.

This document shows how to compile LwIP for using it on a Cypress FM microcontroller using the FM3 type MB9BFD18T on an SK-FM3-176PMC-ETHERNET v2.0 evaluation board, however a similar software example is available for Cypress FM4 microcontrollers as well, using the same low-level driver.

Figure 1. Demo-website running on FM3



2 Hardware Overview

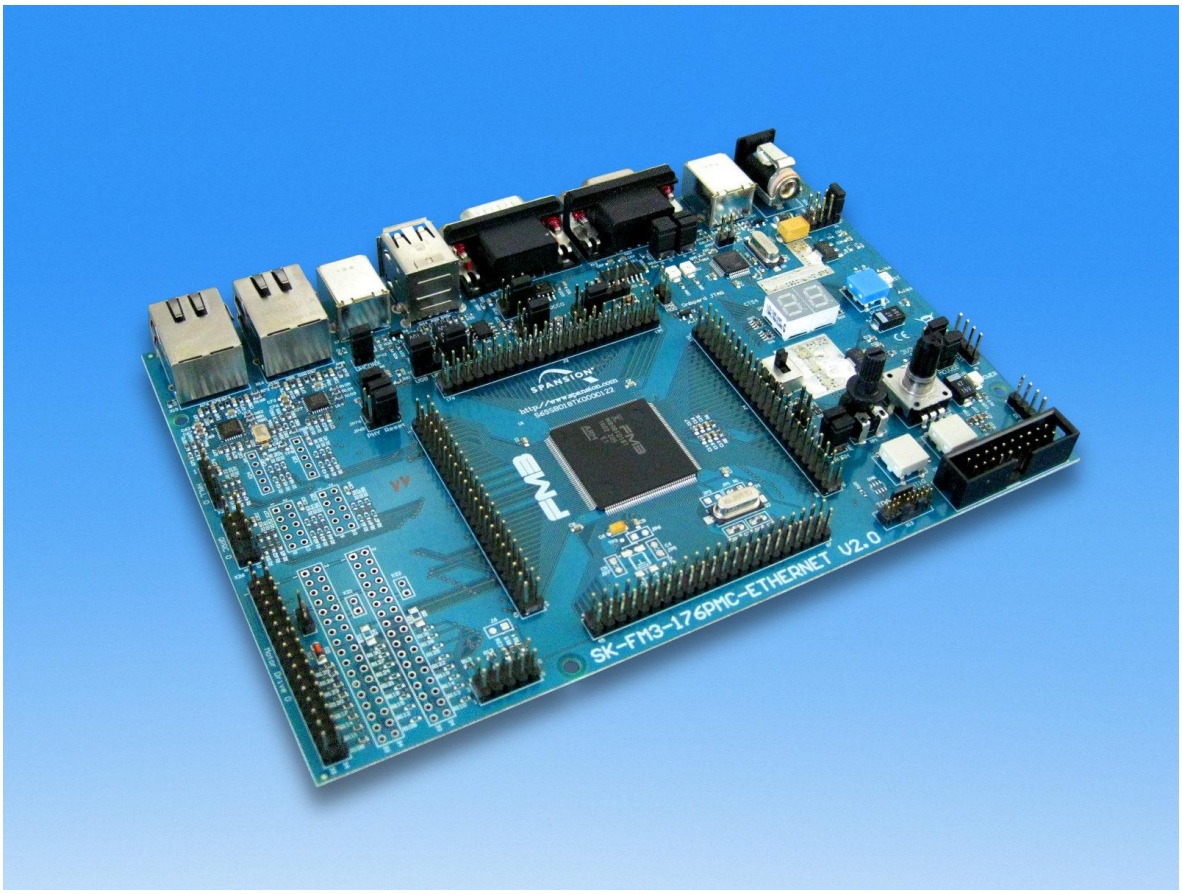
This application note describes the LwIP port to the FM family implemented on an SK-FM3-176PMC-ETHERNET starter kit. In order to understand how and why some functions are implemented the way they are, the hardware is explained briefly in this chapter.

The starter kit SK-FM3-176PMC-ETHERNET uses a Cypress FM3 microcontroller of the type MB9BD10T. It brings 1 MB flash memory, 128 Kbyte RAM and runs at 144MHz CPU frequency.

The demo described here supports the following of the starter kit's features:

- Both Ethernet interfaces can be accessed simultaneously
- Pushbuttons to change value shown on seven segment displays
- Potentiometer to change analog voltage, which is connected to ADC channel 30
- Rotary switch to interface the QPRC module on the MCU

Figure 2. Evaluation board SK-FM3-176PMC-ETHERNET



In order to try out the demo software, please supply the starter kit with power, download the compiled image into the MCU's flash memory and start execution.

Now the LED display should show "00". By pressing the pushbuttons, this value should increase or decrease respectively.

You can connect the board with an Ethernet cable to your PC or a local network. The left Ethernet jack (ETH0) is configured to the static IP address 192.168.1.20, whereas the right one (ETH1) uses DHCP. Some additional hints how to set up your system in order to communicate with the starter kit can be found in chapter 4.1

3 The LwIP implementation on FM Family

This chapter gives a brief overview about some important aspects of this FM port.

3.1 Which files are used

LwIP is a popular open-source TCP/IP stack with an active user community. It aims to have a feature complete external interface and supports among other protocols IP, ICMP, ARP, TCP, UDP and DHCP.

The official project website can be found at http://savannah.nongnu.org/projects/lwip/.

LwIP is shipped in two packages. *lwip* contains the TCP/IP stack and is the official project, whereas additional code is included in *contrib*. The latter brings implementations for services like http, netio, echo, sntp and others.

Depending on your requirements, different combinations of source code files are necessary to be included into your build project. There are four groups of files needed in any case: API, Core, including its subfolders ipv4 and/or IPv6, netif and apps.

This example uses the raw API, so only err.c and tcpip.c are needed. If netif or sockets API are desired, the respective files have to be referenced as well. It does no harm though to include all *.c files inside the api folder as they are not compiled due to preprocessor directives as long as they are not explicitly activated in lwopts.h nor opt.h.

Likewise, all files from the core directory can be integrated. This example does not use dns.c, so the symbol LWIP_DNS is not defined and thus does not consume any resources on the FM3.

Those files realize the TCP/IP functions in LwIP. Applications like web servers have to be added to do anything useful.

A further component is momentous to make the LwIP stack work: The platform specific adaption layer which connects LwIP with the actual hardware drivers. The file ethernetif.c acts as a template for such an interface. This example uses a modified copy of it, which located in the /fm3_adaption folder.

Additionally, lwipopts.h is needed to configure parameters ranging from feature activation to buffer sizes.

3.2 The LwIP adaption layer

LwIP offers two different ways of being used, depending whether the symbol *NO_SYS* is defined or not. This example does not use an operating system and thus has NO_SYS defined to 1. Therefore a small adaption layer is sufficient which consists of the file *ethernetif.*c and connects the stack with the low-level driver.

Otherwise, an operating system emulation layer, consisting of cc.h and sys_arch.c is needed. Please refer to the official documentation for more information[1].

In ethernetif.c, functions for initialization, input and output are implemented which are connected to the LwIP's representation of a network interface by calling the function netif_add() as shown in chapter 4.1.

4 Exploring and developing with LwIP on FM3

This chapter explains how to actually use this demo and gives some practical advice.

4.1 Setting the IP address

```
#if ((LWIP_DHCP) && (DHCP_ETH0 == L3_ON))
IP4_ADDR(&ipaddr, 0, 0, 0, 0);
IP4_ADDR(&netmask, 0, 0, 0, 0);
IP4_ADDR(&gw, 0, 0, 0, 0);
netif_add(&netif0, &ipaddr,&netmask, &gw,(void*)(&EMAC0), &ethernetif_init,
&ethernet_input);
netif_set_default(&netif0);
dhcp_start(&netif0);
#else // static IP address
IP4_ADDR(&ipaddr, 192, 168, 1, 20);
IP4_ADDR(&netmask, 255, 255 , 255, 0);
IP4_ADDR(&gw, 192, 168, 1, 1);
netif_add(&netif0, &ipaddr, &netmask, &gw, (void *)(&EMAC0), &ethernetif_init,
&ethernet_input);
netif_set_default(&netif0);
netif_set_up(&netif0);
#endif // DHCP or static IP address
```

In lwip.c, the function lwip_init() sets up the IP addresses for both interfaces. The code above exemplifies the function at interface 0; if the symbol DHCP_ETH0 is set to L3_ON, the starter kit requests an automatic IP address from a DHCP server. If DHCP_ETH0 is defined to L3_OFF, ETH0 is configured to a static IP address, 192.168.1.20.

[1] doc/sys_arch.txt, http://lwip.wikia.com/wiki/Porting_for_an_OS and http://lwip.wikia.com/wiki/Porting_for_an_OS_1.4.0

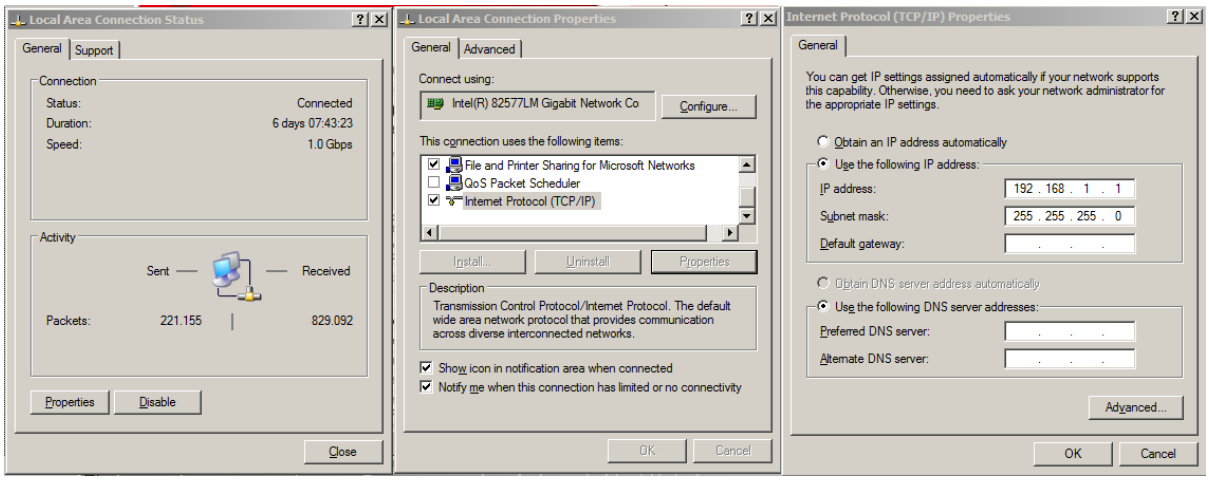
4.1.1 Static address

To connect another Ethernet device with the demonstration package, you have to assign a different static IP address in the same subnet. For instance if you want to connect a PC running Microsoft Windows XP, you can go to ‘Settings’ -> ‘Control Panel’ -> ‘Network Connections’ and select the network interface where you connect the starter kit with.

Henceforth click on ‘Properties’, select from the list ‘Internet Protocol (TCP/IP)’ and choose ‘Properties’. In the appearing dialog, please select ‘Use following IP address’ and enter a suitable IP address. If the example is unchanged, any IP address between 192.168.1.1 and 192.168.1.254 will work.

Please note, that both interfaces may not be part of the same subnet, otherwise routing will not work correctly. This is expected TCP/IP behavior and not unusual. If you intent to use a daisy-chain-like topology, every link must represent a separate subnet.

Figure 3. Configuring static IP address on Microsoft Windows



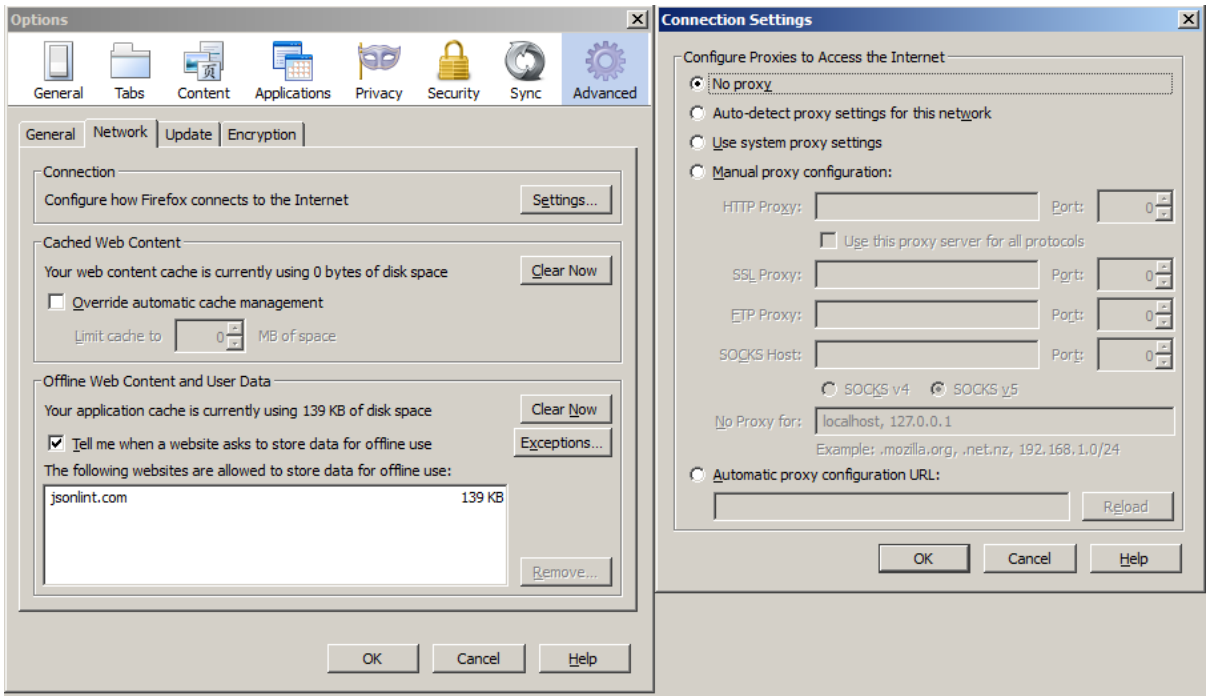
Furthermore, you may have to deactivate any proxy settings

In Mozilla Firefox, go to the menu -> ‘Tools’ -> ‘Options’.

Then select ‘Advanced’, ‘Network’ and click on ‘Settings’.

Here, please select ‘No proxy’.

Figure 4. Proxy configurations in Mozilla Firefox

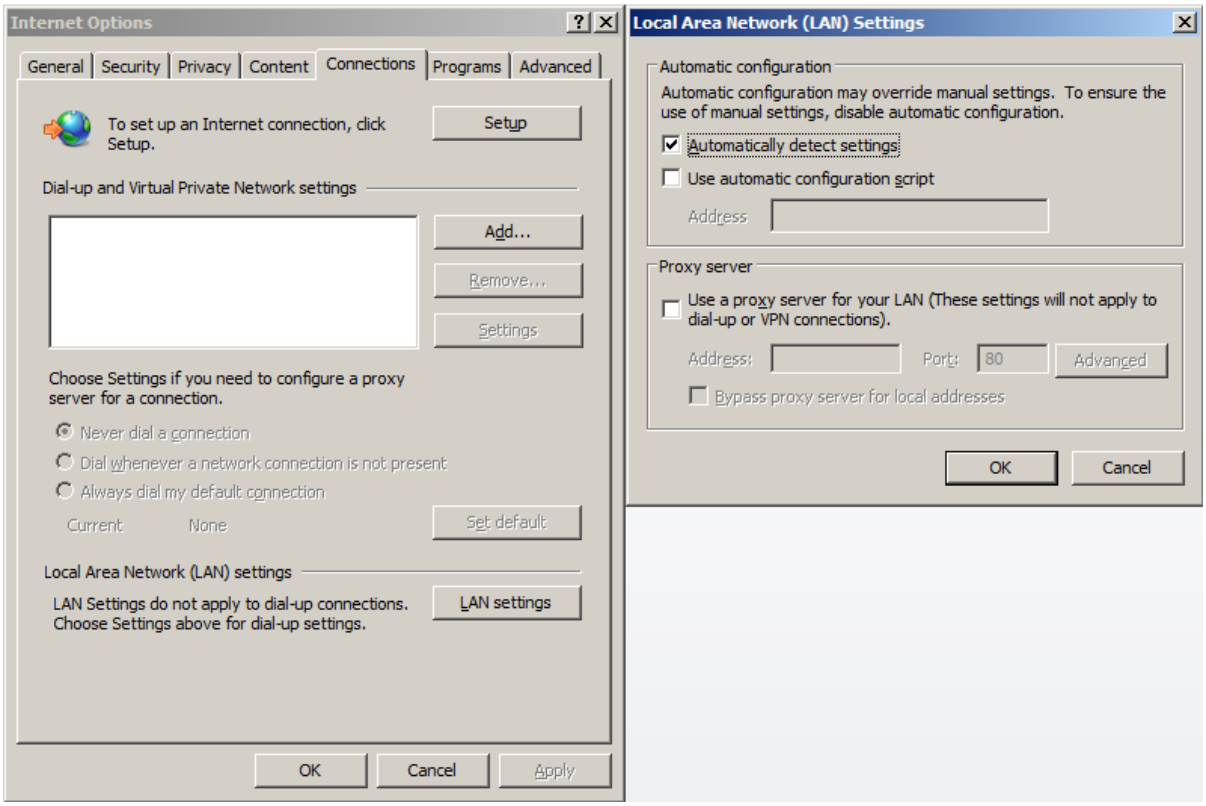


In Microsoft Internet Explorer, proxy settings can be configured like this: Go to the menu -> ‘Tools’ -> ‘Internet Options’.

Then select ‘Connections and click on ‘LAN Settings’.

Here, maybe ‘Automatically detect settings’ must be deactivated.

Figure 5. Proxy settings in Microsoft Internet Explorer 8



4.1.2 DHCP

For DHCP, your Computer and the starter kit must be connected to the same network that also provides a DHCP server. This DHCP server must be configured to accept the starter kit's MAC address, which is defined in the file ethernet_cfg.h like this:

Here, ETH1's MAC address is set to 00:01:01:66:73:38.

```
#define MAC1HWADDR0 (0x00)
#define MAC1HWADDR1 (0x01)
#define MAC1HWADDR2 (0x01)
#define MAC1HWADDR3 (0x66)
#define MAC1HWADDR4 (0x73)
#define MAC1HWADDR5 (0x38)
```

To determine the IP address, you can check your DHCP server or use wireshark[2] (and use on networks with high traffic a filter rule like "eth.addr== 00:01:01:66:73:38") or connect a serial terminal program and read out each interface's link information[3].

[2] Wireshark is a very popular network monitor ("sniffer") tool, formerly known as Ethereal. It is available freely on the Ethernet at http://www.wireshark.org/.

[3] Please refer to section 4.3.1 for instructions on how to use the serial debugging interface.

4.2 Conducting speed measurements

4.2.1 ICMP echo (ping)

On the command line, invoke the ping program with the IP address of your starter kit.

If the TCP/IP stack is configured correctly, the starter kit answers the ICMP requests by sending ICMP responses as depicted in figures 6 and 7.

Figure 6. ping command in Windows

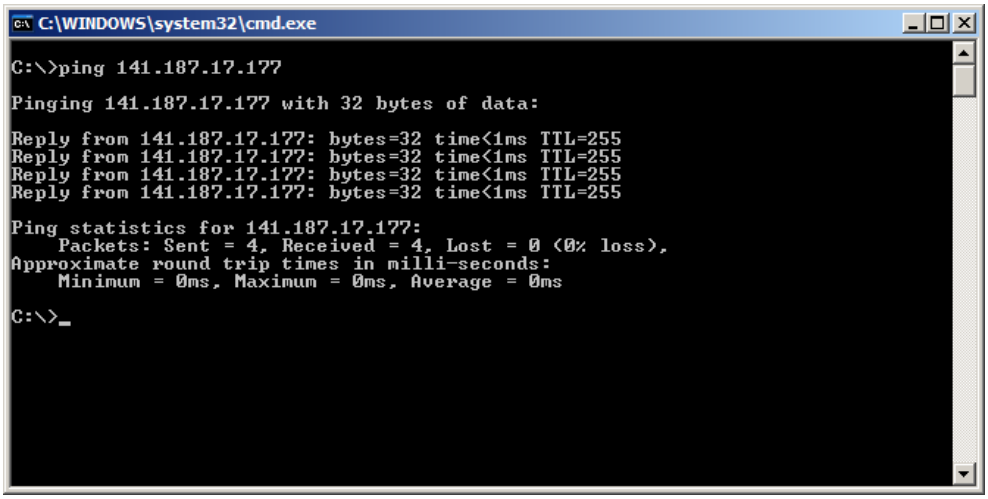
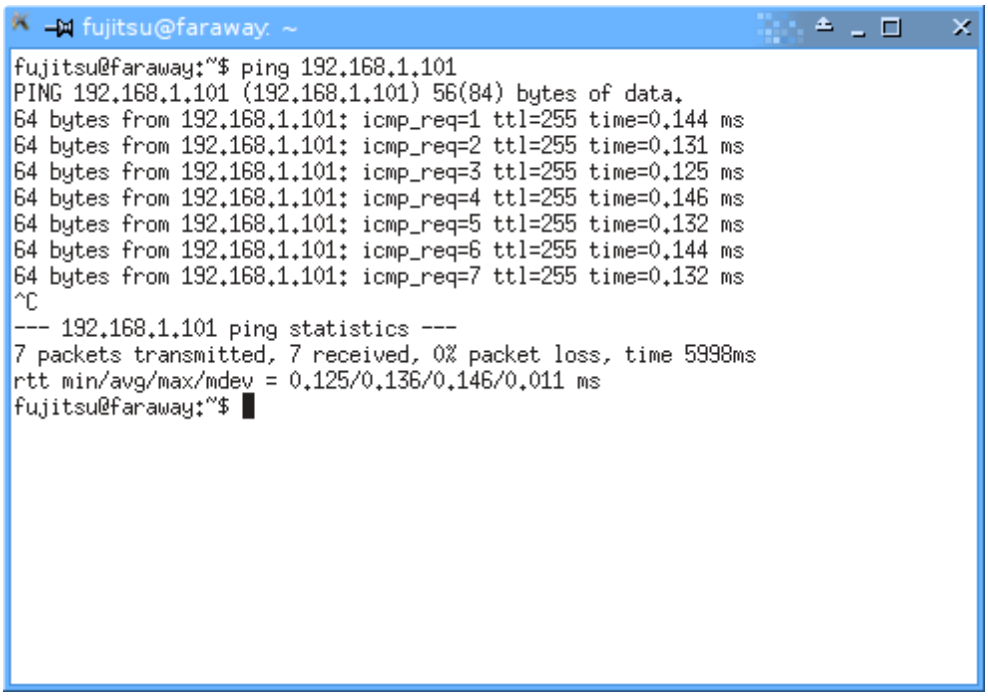
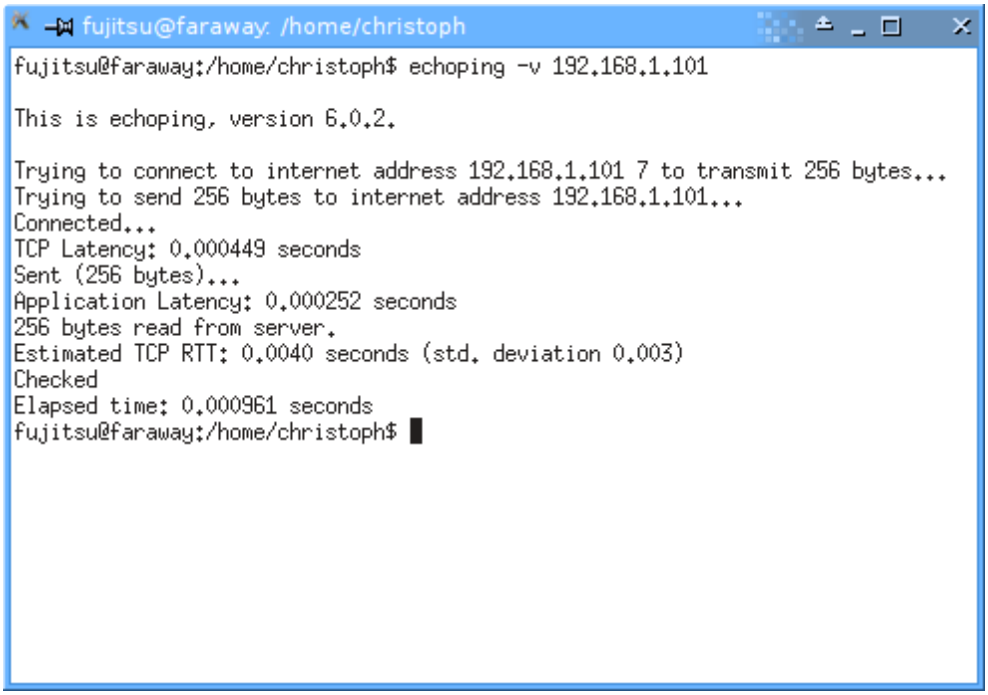


Figure 7. ping command in GNU/Linux



4.2.2 TCP echo

Figure 8. Client for TCP echo server "echoping"



Another facility to test network traffic is the activated “echo server”. This is a service on UDP and TCP port 7, which just send back incoming packets. The purpose is to test if receiving and sending works with those protocols. This is similar as ping but on OSI level four.

4.2.3 NetIO

There is a TCP server for the free network performance benchmark tool *NetIO* by Kai Uwe Rommel. This server is part of the LwIP contrib package. In order to use it, you have to download the NetIO client from http://www.ars.de/ars/ars.nsf/docs/netio and start it with the arguments *-t* to select TCP protocol. The parameter *-b* sets the packet size. As on the small server software only the Tx test is implemented, the client hangs while attempting the Rx measurement and must be terminated by entering CTRL-C.

Figure 9. NetIO with activated Checksum Offload Engine (COE)

```
C:\LocalFiles\Tools\netio\bin>win32-i386 -t -b 32k 141.187.17.200

NETIO - Network Throughput Benchmark, Version 1.31
(C) 1997-2010 Kai Uwe Rommel

TCP connection established.
Packet size 32k bytes:  10.37 MByte/s Tx,
```

You can see the effect of the COE (Checksum Offload Engine) by enabling software checksum calculation in lwipopts.h and repeating this test. You can do that by defining following symbols to 1:

CHECKSUM_GEN_IP, CHECKSUM_GEN_UDP, CHECKSUM_GEN_TCP,

CHECKSUM_CHECK_IP, CHECKSUM_CHECK_UDP, CHECKSUM_CHECK_TCP

Figure 10. NetIO with software-calculated checksums

```
TCP connection established.
Packet size  1k bytes:  4931.61 KByte/s Tx,
```

That means the hardware engine doubles transfer speed compared to the software solution.

4.2.4 HTTP with a webbrowser

There are two websites implemented which you can use for testing performance.

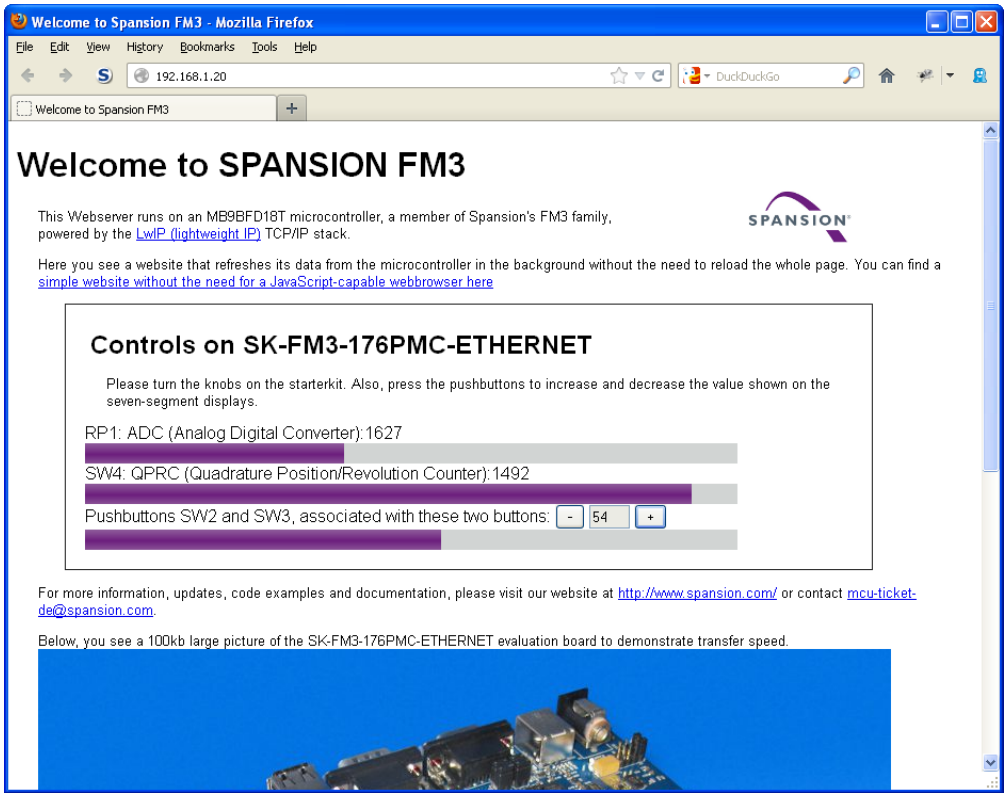
The default page *index.html* is a static website which is sent to the browser unchanged as it is stored in the memory. It contains some JavaScript code, which regularly requests a small data file in the background and changes the respective values of the HTML code. This technique is called *AJAX*[4] and allows websites to contain dynamic content, i.e. data being changed without the need to reload the whole page. AJAX allows the creation of complex *web applications* that can act like desktop applications[5].

The other webpage *simple.shtml* on the other hand is a static website that does not require JavaScript but is created dynamically, i.e. its content changes every time it is reloaded.

The web server that is part of LwIP's contrib-package, decides on the file name extension whether the page to be served is static (.html) or dynamic (.shtml).

Both webpages include an 100kB large image by the name of *bigpicture.jpg*, which is linked to as bigpicture.png?<number> to dissuade the web browser from storing the graphic in its cache memory[6]. To be on the safe side, you can deactivate the browser cache completely.

Figure 11. AJAX-enabled demo-webpage



[4] AJAX: Asynchronous JavaScript and XML – despite the name, other file formats than XML may be used.

[5] With AJAX, even desktop-like user interfaces are possible, like e.g. *eyeOS*: http://www.eyeos.org/

[6] Browsers assume the image to be dynamically created then.

4.2.5 HTTP with wget

the command line tool wget7 is used to download files from the WWW. It displays some statistical data about transfer speed and elapsed time. As for the sake of a speed measurement, we are not interested in the image itself but just in the information how much time is needed to download it, the option --output-document=/dev/null can be added (at least in a POSIX compatible environment like Debian GNU/Linux or the Cygwin tools distribution for Microsoft Windows). This test should be repeated several times to get an idea about the statistical distribution.

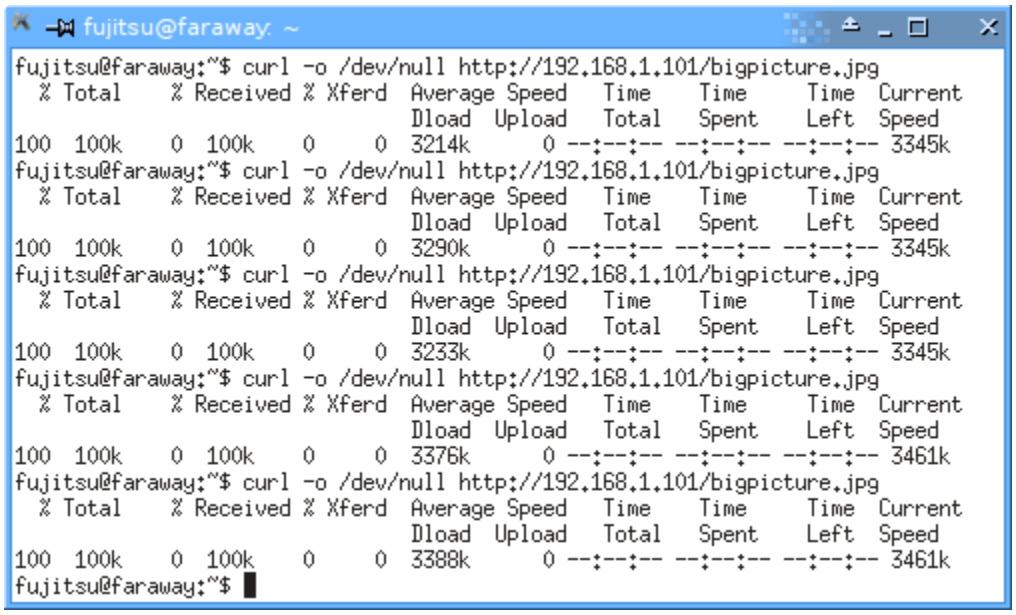
Figure 12. Speed measurement with wget

[7] Available at http://sourceforge.net/projects/wget/

4.2.6 HTTP with curl

The same result can be achieved with the command line tool *curl*[8]. The statistics are different and depending on use case and personal taste, curl or wget is preferred. Here again, multiple program runs should be done to get typical and average figures.

Figure 13. Speed measurement with curl



[8] Available at http://sourceforge.net/projects/curl/

4.3 Debugging utilities

4.3.1 Serial terminal on UART B

This example is configured to use the “UART B” USB interface as output for a virtual serial terminal for printf(). You can access it with a terminal emulator with following settings:

115200 baud, 8 bit, no parity, 1 stop bit and no flow control

You might have to install the device drivers for this virtual terminal first. Please consult your board’s user manual for information how to accomplish this.

4.3.2 LwIP debug options

```
/**
 * LWIP_DBG_TYPES_ON: A mask that can be used to globally enable/disable
 * debug messages of certain types.
 */
#define LWIP_DBG_TYPES_ON               LWIP_DBG_ON

/**
 * ETHARP_DEBUG: Enable debugging in etharp.c.
 */
#define ETHARP_DEBUG                    LWIP_DBG_OFF

/**
 * NETIF_DEBUG: Enable debugging in netif.c.
 */
#define NETIF_DEBUG                     LWIP_DBG_ON

/**
 * PBUF_DEBUG: Enable debugging in pbuf.c.
 */
#define PBUF_DEBUG                      LWIP_DBG_OFF
```

In lwipopts.h, you can activate several options to for LwIP debugging output. The symbol LWIP_DBG_TYPES_ON serves as general switch for this feature. It must be defined to LWIP_DBG_ON if debug messages are desired or LWIP_DBG_OFF otherwise.

All other debug options can be turned on as exemplified above.

4.4 Tweaking memory consumption and performance

Debug functions that write to the serial interface with printf() will slow down the system performance considerably. If it is used synchronously, i.e. not in an operating system, the whole system has to wait until the UART has finished its transmissions on a relatively slow serial link.

The low-level driver can be configured in the file *emac_user.h*. Here you can assign memory space to both Ethernet interfaces according to your needs. Each Ethernet interface has two chains of DMA descriptors, one for reception and one for transmission. Every DMA descriptor has in turn a buffer to hold an Ethernet frame. These parameters must match your expected traffic requirements.

Configure your linker to produce a map file to monitor the overall memory consumption.

LwIP's memory requirements, throughput and latency can be optimized in the file lwipopts.h. The official project wiki discusses this topic in detail. To begin with, please refer to http://lwip.wikia.com/wiki/Lwipopts.h, http://lwip.wikia.com/wiki/Tuning_TCP and http://lwip.wikia.com/wiki/Maximizing_throughput.

As an example, regard the setting MEM_SIZE in lwipopts.h.

If it is dimensioned to small (e.g. 2KiB), the website will build up rather sluggishly:

```
#define MEM_SIZE                        (2*1024)
```

If set to 4KiB on the other hand, the performance is acceptable:

```
#define MEM_SIZE                        (4*1024)
```

4.5 Further documentation on LwIP

LwIP is a popular open-source software with an active user community. LwIP's official project website can be found at http://savannah.nongnu.org/projects/lwip/

You can find a lot of information on the official mailing list lwip-users. For following or participating in current discussions, you can subscribe at http://savannah.nongnu.org/mail/?group=lwip.

Older conversations can be searched in the archive to be found at the URL http://lists.gnu.org/archive/html/lwip-users/. Furthermore there is a wiki online at http://lwip.wikia.com/wiki/LwIP_Wiki.

There is another mailing list addressing the further development of LwIP called lwip-devel whose archive can be accessed at http://lists.nongnu.org/archive/html/lwip-devel/.

The first document to be read when beginning own development should certainly be the README file that comes with lwip. Here is summarized the most important information about the current status of the project including locations of further documentation.

4.6 Modifying websites

The webserver stores the files to be served (html documents, images, css, js …) not natively but converted into a C array inside *fsdata.c*. This file can be generated by calling the converter program *makefsdata.exe* in the path *example/source/lwip1_4_0/app/httpserver_raw*. It by default takes every file located in the subfolder *fs* and overwrites *fsdata.c* with a new version.

So, in order to import your own websites, replace the files in fs with your own and run makefsdata.exe.[9]

After compiling the whole project and flashing it into the FM3, your custom websites should be shown.

The example code shows the usage of SSI (Server Side Includes) and CGI (Common Gateway Interface), which are needed for dynamic content. For more information please refer to the comments in *httpd.c*.

In older versions of Microsoft Internet Explorer the AJAX example may not work without providing an implementation of the JSON object. There will appear an error message stating "'JSON' not defined". A public domain JavaScript library providing all necessary definitions can be found at https://github.com/douglascrockford/JSON-js/blob/master/json2.js.

[9] If you don't use Microsoft Windows, you can compile the converter program yourself from the provided source code in the subfolder *makefsdata*.

To save space in the FM3 microcontroller it is recommended to *minify* this and any larger JavaScript file, e.g. with a program available at http://javascript.crockford.com/jsmin.html. It removes comments and for correct function unnecessary whitespaces. json2.js's memory consumption is reduced from 16KB to about 4KB – for an embedded system a considerable amount.

5 More information about FM Family and support

5.1 Overview about Cypress FM microcontroller family

All information about FM Family product line, documentation, tools, news and application examples, you can find at:

http://www.cypress.com/products/32-bit-arm-cortex-mcus

5.2 Hardware tools

An overview of available FM3 evaluation boards is available at: http://www.cypress.com/FM3evaluationboards

Information about the SK-FM3-176PMC-ETHERNET evaluation board, which this application note is based upon, can be found at: http://www.cypress.com/SK-FM3-176PMC-ETHERNET

5.3 Software tools

To download compiled firmware files into the FM microcontroller's internal flash memory, you can use the Flash MCU Programmer for FM0+/FM3/FM4 or the Flash USB Direct tool: http://www.cypress.com/flashsoftwaretools

5.4 Software examples

You can download the newest version of this and other example projects from the starter kit's or microcontroller's website, such as: http://www.cypress.com/SK-FM3-176PMC-ETHERNET

http://www.cypress.com/SK-FM4-216-ETHERNET

http://www.cypress.com/MB9BFD18TPMC-GE1 or http://www.cypress.com/S6E2CCAJ0AGV20000

6 Document History

Document Title: AN204414 - LwIP over Ethernet on FM Family

Document Number: 002-04414

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	_	CHNO	10/02/2012	CNo, First edition
			12/19/2012	CNo, updated to reflect changes on Ethernet driver
*A	5073184	CHNO	01/13/2016	Converted Spansion Application Note “AN706-00056-2v10-E” to Cypress format as v20 was never official but merely an ad-hoc document.
*B	5874563	AESATMP9	09/06/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709