文章编号：1672-2892(2010)01-0067-04

LWIP 在 TMS320DM642 平台的实现

徐 萍，卿粼波*，何小海，杨 超，许光辉

(四川大学 电子信息学院，四川 成都 610064)

摘 要：TMS320DM642 平台上实现 TCP/IP 通信，可为嵌入式多媒体系统的网络应用提供技术支持。分析 TMS320DM642 芯片功能和轻量级网络协议(LWIP)层次结构，通过 LWIP 的移植，实现 TMS320DM642 非网络开发者套件(NDK)解决方案的 TCP/IP 网络通信，实例验证了 LWIP 移植方法的有效性。

关键词：TMS320DM642 平台；轻量级网络协议；TCP/IP 协议栈

中图分类号：TN915.04；TN915.09 文献标识码：A

LWIP's Implementation based on TMS320DM642 platform

XU Ping，QING Lin-bo，HE Xiao-hai，YANG Chao，XU Guang-hui

(School of Electronics and Information Engineering，Sichuan University, Chengdu Sichuan 610064，China)

Abstract：The implementation of TCP/IP protocol stack on TM320DM642 platform could provide technological support for the application of multimedia embedded systems to networks. Both the function of TM320DM642 chip and the hierarchical structure of Light Weight Internet Protocol(LWIP) were analyzed. Network communications for TM320DM642 systems were implemented by the transplantation of LWIP, not by a solution scheme of Network Developer's Kit(NDK). Testing examples verified the effectiveness of the LWIP transplantation method.

Key words：TMS320DM642；LWIP；TCP/IP protocol stack

TMS320DM642[1]是 TI 公司针对多媒体处理领域应用需求，而重点推出的一款数字信号处理器(Digital Signal Processor, DSP)芯片[2]，集成了音视频和网络通信等外设，在基于网络的音视频传输、数字视频记录、机器视觉、医学成像、安全监视和数字相机等嵌入式系统中得到广泛应用。当前 TMS320DM642 平台的网络实现方案主要是 TI 公司提供的 NDK，但用户需要支付高额的费用才能使用，而且 NDK 源码不公开，不适合于裁剪。

LWIP[3]是瑞士计算机科学院 Adam Dunkels 等开发、用于嵌入式系统的 TCP/IP 协议栈，源代码开放，结构清晰，注释详尽，可移植性、可裁减性好，不仅可移植到操作系统上，也可在无操作系统的情况下独立运行。LWIP TCP/IP 实现重点是在保持 TCP 协议主要功能的基础上减少对 RAM 的占用，一般只需要几十 KB 的 RAM 和 40 KB 左右的 ROM 就可以运行，这使 LWIP 协议栈适合在低端嵌入式系统中使用。将 LWIP 移植到 TMS320DM642 为核心的视频开发系统中，具有一定的实用性和经济性。

1 TMS320DM642 简介

TMS320DM642(以下简称 DM642)最高主频为 720 MHz，通用主频 600 MHz，每秒最多可完成 4.8 G 次操作。DM642 具有 64 bit 的外部存储接口(Eternal Memory Interface, EMIF)，可以实现与异步存储器和同步存储器的无缝连接，同时支持 1024 MB 的可编址外部存储空间，它还配备了 10/100 Mb/s 以太网媒体访问接口控制器(Ethernet Media Access Controller, EMAC)、I2C 总线、3 个可配置的视频口(Video Port, VP)、多通道音频串口和两个多通道缓冲串口。DM642 平台主要功能模块包括视频采集显示模块、网络传输模块、外围芯片扩展模块、JTAG(Joint Test Action Group)调试模块等，其功能结构图如图 1 所示。

收稿日期：2009-07-24；修回日期：2009-09-04

基金项目：教育部科学技术研究重点资助项目(107094)

*通信作者：qing_lb@scu.edu.cn

2 LWIP 的移植

2.1 LWIP 简介

LWIP 协议栈，在可移植性、可集成性，以及平台无关性上有着卓越的性能[3-5]。它具有以下特性：支持多网络接口下的 IP 转发，支持 ICMP 协议，包括实验性扩展的 UDP、阻塞控制、RTT 估算、快速恢复和快速转发的 TCP，提供专门的内部回调接口，并提供了可选择的 Berkeley 接口 API。与通常的协议栈不同，LWIP 并不是采用 TCP/IP 协议的每一层作为一个单独的进程，而是把所有的 TCP/IP 协议栈放在一个进程当中，这样 TCP/IP 协议栈和操作系统内核分开，也避免了数据跨层时带来的上下文切换。

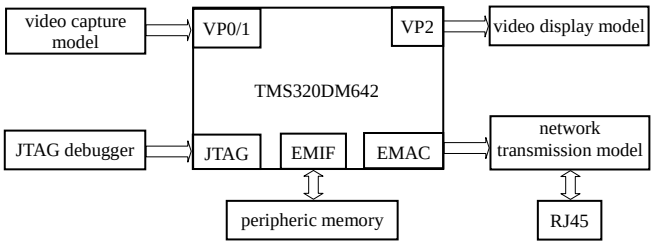

Fig.1 DM642's main function model
图 1 DM642 主要功能模块

基于 LWIP 的应用系统总体结构层次可分为 3 层，即：应用层、TCP/IP 协议层、操作系统模拟层，如图 2 所示。其中，操作系统模拟层是整个系统的最底层，通过它，协议栈能够调用操作系统信号量、邮箱、线程阻塞等功能，对协议栈的移植者而言，也只须把主要精力集中在对操作系统模拟层的实现上，从而忽略 TCP/IP 协议栈的真正实现细节。

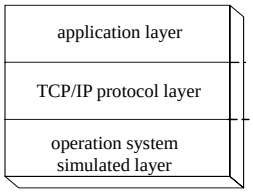

Fig.2 Hierarchy of LWIP application system
图 2 LWIP 应用系统层次结构图

2.2 LWIP 的移植准备工作

LWIP 移植的准备工作主要包括：下载 LWIP 源码；建立基于 BIOS 的工程，为下载的 LWIP 源码新建目录，将目录名命名为 Lwipsrc，将源码解压后放入 Lwipsrc 目录；将目录中的源文件加入工程，包括/api 目录下的所有源文件、/netif 下的 earthnetif.c 文件、/core 目录下的所有源文件以及/core/ipv4 下的源文件；编译工程，去除编译错误。

初编译时错误较多，原因可能是缺少一些头文件，包括与操作系统设置相关、与处理器相关的一些头文件，以及配置 LWIP 的一些自定义头文件。

在 opt.h 中，包括一项#include "lwipopts.h"，主要用于包含移植者自定义的头文件 lwipopts.h，在本测试系统中没有自定义的头文件，因此，注释掉该语句，减少编译错误。

2.3 系统模拟层的实现

如前所述，系统模拟层向协议层提供操作系统的服务，使协议层不必关心底层实现的细节。根据 LWIP 协议栈的文档，系统模拟层需要提供邮箱、信号量、定时器、进程建立与删除等操作系统功能。模拟层的实现可分为两个步骤进行：

1) LWIP 协议栈为兼容不同处理器架构与编译器平台的数据类型，定义了与处理器无关的数据类型。为将 LWIP 协议栈的数据类型与本文所用的 DM642 平台关联起来，本文定义了 cc.h。文件定义如下：

```
typedef unsigned char    u8_t;
typedef signed char    s8_t;
typedef unsigned short    u16_t;
typedef signed short    s16_t;
typedef unsigned int    u32_t;
typedef signed int    s32_t;
```

2) 编写与操作系统模拟层相关的一些结构和函数，主要涉及如下函数：

a) sys_sem_t 信号量

LWIP 中需要使用信号量功能，而 DSP/BIOS 提供了类似的信号量功能函数。因此，除 sys_arch_sem_wait() 以外，只要把 DSP/BIOS 的信号量操作的函数重新封装成 LWIP 所需的函数，就可实现信号量操作。sys_arch_sem_wait()函数中通过调用 SEM_pend()来实现。但 SEM_pend()返回的是在约定 timeout 时间内，是否等到信号量的 bool 值，而在 LwIP 中 sys_arch_sem_wait()的返回值是等待信号量有效所花费时间的毫秒数，所以需要调用 DSP/BIOS 中的 CLK 模块中的 CLK_getltime()函数来获得低分辨率时间。另外 CLK_getltime( )函数所返回的时间指的是与时钟中断的次数，与 BIOS 中时钟模块的设置有关，因此需要另外定义一个宏 TICK_TO_MS 完成中断次数与毫秒数的转换。

下面给出了 sys_arch_sem_wait()函数的源代码：

```
u32_t sys_arch_sem_wait(sys_sem_t sem, u32_t timeout)
{
   u32_t i,starttime,endtime;
   starttime=CLK_getltime(); //得到 PEND 前的 TICK 值
   i=SEM_pend(sem, (timeout ? MS_TO_TICK(timeout) : SYS_FOREVER) );
   endtime=CLK_getltime();//得到 PEND 后
   if(FALSE==i){ return ( SYS_ARCH_TIMEOUT ); }
   else{return TICK_TO_MS(( endtime-starttime)); }
}
```

b) sys_mbox_t 消息

LWIP 使用消息队列来缓冲、传递数据报文，同样地，DSP/BIOS 用邮箱(mail box)实现了消息队列结构及其操作，其实现方法基本跟 sys_sem_t 信号量的操作函数类似，只需将 DSP/BIOS 重新进行封装即可。

c) sys_arch_timeout 函数

在 TCP/IP 协议中经常要用到定时，LWIP1.10 以后的版本改进以往需要借助被移植平台时钟中断产生定时事件的定时实现机制，而巧妙地采用定时事件链表的方式，借用 sys_arch_sem_wait(),sys_arch_mbox_fetch()等导致 CPU 阻塞的函数产生定时事件。

在 LWIP 中每个和 TCP/IP 相关的线程都有一个定时事件链表，为此需要为每个线程分配一个链表，并在线程建立时将定时时间链表与线程相关起来。

d) sys_thread_new 函数

sys_thread_new()函数创建一个线程，用 DSP/BIOS 中的任务创建函数 TSK_create()可以很方便地实现线程的创建。值得注意的是须同时创建对应该线程的事件链表，并将线程句柄与事件链表关联起来。

e) sys_init 函数

用以对操作系统模拟层进行初始化，主要是设置事件链表参数，初始化线程事件数据，将线程与事件链表值的初值赋为空。

2.4 物理层接口实现

在 LWIP 中可以有多个网络接口，每个网络接口都对应了一个 netif 结构，它包含了相应网络接口的属性、收发函数。LWIP 调用了 netif 的方法 netif->input()及 netif->output()进行以太网 packet 的收、发等操作。在驱动中主要完成的就是实现网络接口的收、发、初始化以及中断处理函数，与网络接口设备驱动相关的函数定义在 ethernetif.c 文件中。

驱动程序工作在 IP 协议模型的网络接口层，它提供给上层(IP 层)的接口函数如下：网络初始化函数，ethernetif_init()；网络接收函数，ethernetif_input()；网络发送函数，ethernetif_output()。

DM642 的 CSL 库提供了对片上 MDIO 和 EMAC 模块的操作接口，参考文献[4]给出了基于 CSL 的以太网 RAW 数据参考例程。CSL 提供了包发送的函数，EMAC_sendPacket()；对于数据包的接收函数，CSL 则是通过注册回调函数实现的。通过在包接收中断中调用包接收回调函数完成数据包的接收操作。TI 提供的 EMAC 编程参考模型，将对包缓存的管理交给了基于 CSL 的 API 以上的部分。CSL 另外还预留了用于数据管理的包获取以及包释放回调函数接口。对于包接收，由于 CSL 库是通过回调函数进行处理的，即包接收的处理是在中断中执行的，协议栈包接收处理的过程比较复杂，占用 CPU 的资源较多，这样将影响系统的实时性。因此，本文在线程分配的时候，尽可能减少在回调函数中的包处理，而新设一线程 thrRxpacket，完成包的处理。当接收到包的时候，中断程序调用回调函数，而回调函数将包封装成 SCOM 消息，发送到 thrRxpacket，由它完成将包组成 LWIP 协议栈接收的 pbuf 格式数据包，并完成包预处理的工作，然后将包封装成消息送到 LWIP 协议栈主线程进行处理。

3　性能测试

将本平台通过以太网与 PC 机相连，由于数据包的发送与接收基本类似，本文仅分别测试了平台发送 UDP 包、发送 TCP 包的性能。本文采用了周立功公司开发的“TCP&UDP 测试工具 1.025”进行测试。

在平台中分别添加 TCP 和 UDP 的测试线程：thrTestUdpSend,thrTestSendTcp，在线程中添加对 LWIP 功能调用的 API 代码调用。

TCP 线程的测试代码如下：

```
void thrTestSendTcp()
{
   struct netconn *conn, *newconn;
   conn = netconn_new(NETCONN_TCP);   //建立一个新连接
   netconn_bind(conn, NULL, 6000);    //绑定到 6000 端口
   netconn_listen(conn);              //网络连接设置为侦听
   while(1) {
            newconn = netconn_accept(conn); //等待接收连接请求
            netconn_send(newconn,buf,size);  //发送数据包，buf 里的值设置为 0xff
            TSK_sleep(1); //通过设置睡眠的时间，调控数据包发送的频率，以调整数据发送速率。
         }
}
```

UDP 线程的测试代码与之类似，在此不作赘述。测试结果如下：使用 ping 命令连接主机，如图 3 所示；测试工具测试发送数据包速度的屏幕截图如图 4；测得 TCP 发送数据包与 CPU 占用率关系如表 1；测得 UDP 发送数据包与 CPU 占用率关系如表 2。其中，PC 机 IP：192.168.1.42；平台 IP：192.168.1.40。

表 1 用 TCP 发送数据包 CPU 占用率与数据传输率表

Table1 CPU occupancy rate and data transmitting rate by TCP (TCP package size is set 512 byte)

CPU occupancy rate /(%)	2.38	4.69	6.63	9.37	18.45
data transmitting rate /(Mbit/s)	0.19	0.38	0.58	0.80	1.51

表 2 用 UDP 发送数据包 CPU 占用率与数据传输率表

Table2 CPU occupancy rate and data transmitting rate by UDP (UDP package size is fixed and set 1 000 byte)

CPU occupancy rate /(%)	2.74	5.43	8.85	11.73	91.47
data transmitting rate /(Mbit/s)	0.51	1.02	1.61	2.22	2.76

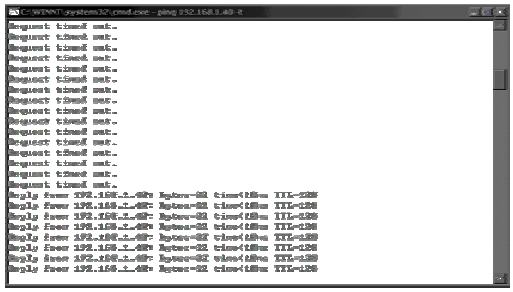

Fig.3 Using ping order to connect host computer
图 3 使用 ping 命令连接主机

Fig.4 Test data transmitting speed
图 4 数据速率测试屏幕截图

4 结论

在 DM642 嵌入式平台上实现了 LWIP 的移植，根据实际需要裁剪了其代码，并通过实例测试了 LWIP 的性能，使得基于此平台的图像网络传输成为现实，而不必向 TI 付高额的费用购买 NDK 网络解决方案。

参考文献：

[1] Magdalena Iovescu,Mike Denio. Software Operation of Gigabit Ethernet Media Access Controller on TMS320C645x DSP[Z]. Texas Instruments, 2006.
[2] 彭启综,李玉柏,管庆. DSP 技术的发展与应用[M]. 北京:高等教育出版社, 2002.
[3] Adam Dunkels. Design and Implementation of the LWIP TCP/IP Stack[R]. Stockholm:Swedish Institute of Computer Science, 2001.
[4] 焦海波. uC/OS-II 平台下的 LwIP 移植笔记[EB/OL]. (2009-04-30)[2009-07-24]. http://group.ednchina.com/641/24300.aspx.
[5] Doudlas E Comer. 用 TCP/IP 进行网际互联第一卷:原理、协议与结构[M]. 林瑶, 等译. 北京:电子工业出版社, 2001.

作者简介：

徐 萍(1985-)，女，四川省眉山市人，在读硕士研究生，主要研究方向为图像处理与图像通信.email:qing_lb@scu.edu.cn.

许光辉(1984-)，男，河南省商丘市人，在读硕士研究生，主要研究方向为电路与系统.

卿粼波(1982-)，男，四川省资阳市人，博士，讲师，主要研究方向为图像处理与图像通信.

何小海(1964-)，男，四川省绵阳市人，博士，教授，主要研究方向为图像处理与图像通信.

杨 超(1984-)，男，四川省绵阳市人，硕士，主要研究方向为图像处理与图像通信.