基于 μClinux 的网络构件的设计与实现

周 武，肖 军
（西安航空技术高等专科学校 电气工程系，陕西 西安 710077）

摘 要：主要介绍 32 位微处理器 S3C44B0X 和嵌入式操作系统 μClinux 的开发和研究，提出了 ARM 和 μClinux 实现以太网通信的一种方案，即一套基于嵌入式操作系统的嵌入式网络软件开发平台，在此平台之上。可以方便进行嵌入式应用系统的开发。

关键词：嵌入式操作系统；μClinux；ARM；移植
中图分类号：TP316.8 **文献标识码**：A

Design and realization of network structure based on μClinux

ZHOU Wu, XIAO Jun
(Department of Electrical Engineering, Xi'an Aerotechnical College, Xi'an 710077,China)

Abstract: This paper mainly introducts development and research of the 32 bit ARM micro-controller S3C44B0X and the embedded operating systems μClinux.This paper puts forward solution of the embedded system access to the network is actually a network software platform based on an embedded real-time operating system, and applications development based on this platform must be convenient.

Key words: embedded operating system; μClinux; ARM; transplant

随着网络技术的发展，在工业监测、控制等各个领域，嵌入式系统将越来越多地支持互联网功能。人们对互联网的嵌入式系统的功能和可靠性都提出了越来越高的要求。同时，随着微电子技术和半导体技术的迅速发展，高档处理器的成本大幅度下降及软件技术的发展，嵌入式操作系统日趋成熟。

本文对基于 32 位微处理器和 μClinux[1]的嵌入式系统进行了研究，从硬件核心—32 位微处理器的选型，其他硬件部分的设计，系统驱动程序的编写，嵌入式操作系统和文件系统的移植，完成了对一个嵌入式系统从计划设计到开发调试的完整过程的研究。

1 网络构件的整体结构设计

μClinux 的网络构件的硬件结构图如图 1 所示。本系统采用韩国三星公司的 S3C44B0X[2]微处理器,外扩一片 norflash 芯片 HY29LV160B、nandflash 芯片 K9F2808u 和 SDRAM 芯片 HY57V561620，JTAG 接口，RS232 串口；带有开关量采集模块、模拟量采集模块和以太网通信模块；采用 μClinux 嵌入式操作系统来进行软件设计。

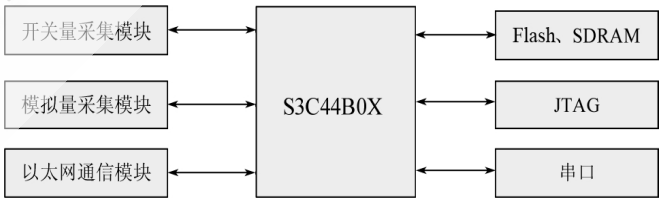

图 1 硬件结构框图

硬件系统上使用了两种类型的 Flash，一种是 norflash，另一种是 nandflash。norflash 适宜连续大容量存储，价格相对便宜；相比较，nandflash 随机存储速度快、价格高。所以在本系统中结合两种 Flash 的优势，将移植的操作系统存放在 norflash 之中，nandflash 则是用来存储应用程序的代码和常量，保证用户的程序在掉电后不丢失。

该方案设计相对简单，硬件电路中采用韩国三星公司的 S3C44BOX 微处理器，8 KB Cache、可选的内部 SRAM、2 通道 UART、8 通道 10 bit ADC、71 个通用 I/O 口、2 个可编程 32 bit 定时器，能够基于芯片设计复杂的系统。其架构满足了 μClinux 正常运行的基本要求。

2 系统软件设计

为使该系统具有较好的实时性和稳定性，在μClinux平台上设计系统软件。系统中各个任务在宏观上按照一定的关系并行工作，CPU资源得到充分利用，系统可靠性得到很大的保证，方便组织开发任务。在μClinux平台上，软件设计工作主要包括：Bootloader的移植、μClinux在S3C44B0X上的移植、驱动程序的编写和应用程序的编写。

2.1 Bootloader的移植

Bootloader是嵌入式系统软件开发的第一个环节，它紧密地将软硬件衔接在一起，对于一个嵌入式设备后续的软件开发至关重要。Blob是Boot Loader Object的缩写，是一款功能强大的Bootloader。MBA44B0是一款基于S3C44B0的开发板。本文将以运行在MBA44B0开发板上的Blob的源代码为基础，再针对自己的开发板进行Blob的移植。Blob的启动流程的文件关系如图2所示。

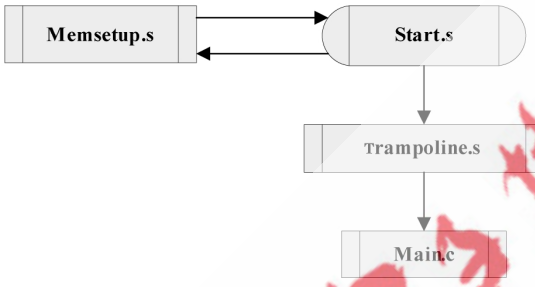

图2 Blob的启动流程的文件关系图

Blob编译后的代码定义最大为64 KB，并且这64 KB又分成两个阶段来执行。第一阶段的代码在start.s中定义，大小为1 KB，它包括从系统上电后在0x00000000地址开始执行的部分，并运行在Flash中，包括对S3C44B0的一些寄存器的初始化和将Blob第二阶段代码从Flash拷贝到SDRAM中。余下63 KB代码都是第二阶段的代码。其起始文件为Trampoline.s，被复制到SDRAM后，就从第一阶段跳到这个文件开始执行剩余部分代码。这个阶段最大为63 KB，单词trampoline词义为“蹦床”，所以在这个程序中进行一些BSS段设置、堆栈的初始化等工作后，最后跳转到Main.c进入C函数。

2.2 μClinux的移植[3]

μClinux2.4.x发行包中的内核对S3C44B0X处理器的支持是不完整的，因此，不可能在make config配置选项中选中S3C44B0X目标板后，直接编译它来得到一个很好的支持S3C44B0X开发板的内核映像。这里对μClinux原代码的改写主要是对网卡支持部分和中断部分（这部分时为以后扩展准备的），由于RTL8019AS网卡与NE2000系列的网卡是兼容的，所以可以直接借助μClinux\drivers\net\ne.c源代码进行改写。

（1）在Ne.c中函数ne_probe就是网卡的检测函数，如果检测到Ne2000兼容的网卡就是return 0，可以参考一下添加的函数的语法格式，将网卡的基地址、中断号都放到这里面定义：

```
#elif defined(CONFIG_ARCH_S3C44B0)
    static int once = 0;
    if (once)
    return -ENXIO;
    if (base_addr == 0) {
    dev->base_addr = base_addr = ARM_NE2000_BASE;
    dev->irq = ARM_NE2000_IRQ;
    once++;
}
```

其中，ARM_NE2000_BASE和ARM_NE2000_IRQ是在配置内核的时候定义的。

（2）ne_probe是被Space.c调用的，这里网卡的检测是从./drivers/net/Space.c的ethif_probe函数中实现的，关键代码如下：

```
    if (probe_list(dev, eisa_probes) == 0)
      return 0;
    eisa_probes                  : 在前面定义成全局
    static struct devprobe eisa_probes[] __initdata = {
    #ifdef CONFIG_DE4X5???????????? /* DEC DE425,
DE434, DE435 adapters */
      {de4x5_probe, 0},
    #endif
    ……
    {NULL, 0},
    };
```

添加的函数是：

```
      if (probe_list(dev, arm_probes) == 0)
      return 0;
```

并定义：

```
    static struct devprobe arm_probes[] __initdata = {
    #ifdef CONFIG_ARM
    {ne_probe, 0},
#endif
    {NULL, 0},
```

（3）地址偏移的问题

同样是在ne.c中ne_probe1的代码中。为了更好地说明所修改的地方，首先应该先介绍一下硬件的配置和连接。这里8019在S3C44B0的Bank 5上，工作在跳线模式，所以起始基地址就是0x0a000600。还有一点需要特别注意的是：8019工作在16位模式下，数据线一对一地连接，地址线错开一位，即8019的A0连接S3C44B0的A1……这样，8019的基地址（Reg0的地址）是0x0a000600，Reg1的地址就是0x0a000602……所以地址不是连续增加的，那么对应的驱动程序要做相应的修改。

（下转第35页）

CHMRecFromSDUTime (1)

该差值可以认为是本次 CHM 处理该测试帧的时间，也可以认为是 CHM 处理下一个测试帧的时长（ms 级）。CHM 需要将该数据通过相关接口告诉 SDU 模块，便于正确产生测试帧。该接口包括的主要类型：CHM 当前的处理时长和信道板收到测试帧的 GPS 时刻。对于信道板处理时长字段，可以和 SDU 协商使用一个字节来标识：其中 2 bit 的步长域处理延时的步长；其余 6 bit 指出延时相对于步长的表示。CHM 处理时长的表示方式如表 1 所示。

表1 CHM处理时长的表示方式

步长域取值	时间步长/ms	表示范围/ms
00	125	±3.875
01	1.0	±31.0
10	2.5	±77.5
11	保留	保留

在 SDU 解析该字段时，可以按照表 1 的对应关系，使用下面的公式：

CHMProcessTime = Scale × Multiple (2)

该结构中还有另一个字段用于标识 CHM 收到 SDU 测试帧的 GPS 时刻，由于 SDU 知道自己产生该测试帧的 GPS 时刻，计算两个时间的差值可以得到 SDU 与 CHM 之间的链路延迟时间 LinkDelay：

LinkDelay=CHMRecFromSDUTime−SDUConstructFrameTime (3)

SDU 根据 CHMProcessTime 与 LinkDelay 的和得到上次发送测试帧的延迟时间，用于修正下一次产生业务帧的时间。对于由 SDU 生成的前向测试帧，SDU 将产生帧的全球定位系统时间与延时相加，就可得知 CHM 发送帧的时间，也就是基站收发信机射频前端处发送 20 毫秒帧的全球定位系统时间；对于产生的反向测试帧，需要将收到来自 MS 的测试帧的 GPS 时间减去该延迟，空间的延迟可以不用考虑，即可得到 MS 产生该帧的时刻。

本文创造性地提供了一种简单的自适应方法来实现 Markov 测试帧的精确同步，目前该方案在中兴通讯 3G 系统网管软件平台的实际应用中表现出了较好的效果。使用该方法解决 Markov 测试帧的同步问题能很好地满足 Markov 测试的需求。

本文针对 Markov 测试帧精确同步问题，在分析传统做法的基础上，提出了一种新的自适应的方法，在进行 Markov 测试时，动态地根据运行环境自动调整链路延时和处理延时，从而保证 Markov 测试帧的精确同步。

参考文献

[1] 3GPP2 C.S0025.Markov Service Option (MSO) for cdma2000 Spread Spectrum Systems.2000.
[2] 3GPP2 C.S0010.Recommended Minimum Performance Standards for cdma2000 Spread Spectrum Base Stations. 1999.
[3] 陈化钧.从3G外场测试看未来网管. 通信产业报，2004(12).
[4] 冉晓明.CDMA 扩频通信.数据通信，1998(3).
[5] 黄永康，傅范丰. 中兴ZXG10-OMC操作维护中心.通讯世界，1999，6(6)：62-63.
[6] ITU-T Recommendatio X700.Management Framework For Open Systems Interconnection (OSI) For CCITT Applications.1992.
[7] 陈小光，陈蔚薇，郭丽丽. 嵌入式软件运行剖面建模及测试用例生成[J].微计算机信息，2008，4(4)：243-245.

（收稿日期：2008-12-14）

(上接第 32 页)

```
#elif defined(CONFIG_ARM)
#define EI_SHIFT(x) ((x)*2)
```

其中 EI_SHIFT 可以查看到 8390.h 的定义。

也有直接访问外部的代码，所以要添加的还有：

```
#ifdef CONFIG_ARM
    regd = inb_p(ioaddr + 0x0d*2);
    outb_p(0xff, ioaddr + 0x0d*2);
#else
    regd = inb_p(ioaddr + 0x0d);
    outb_p(0xff, ioaddr + 0x0d);
```

：函数 outb_p 和 inb_p 访问外部 IO 的函数

这样就被解决了地址偏移的问题，这里采用预处理来添加自己的代码，不直接在原有的代码上修改，可以保证代码的完整性和可移植性，也较容易比较和发现问题。

主程序和 μClinux 中的系统文件放在同一个程序下，进行编译即可。为了提高执行效率，可以根据实际应用修改 μClinux 的部分常用代码，甚至剪切掉某些不必要的代码。

基于 μClinux 的网络构件的设计方案在硬件上简洁可靠；软件可维护性好，可扩展性好，有利于系统的后续开发，降低了系统设计的复杂性。随着嵌入式产品研究的深入，网络接口芯片的研究也会快速发展，使智能化产品的设计更趋向简单、标准、成熟。可以看出，嵌入式 μClinux 操作系统与网络将会得到更大的发展和更广阔的应用。

参考文献

[1] 周立功，陈明计.ARM嵌入式linux系统构建与驱动开发范例[M].北京：北京航空航天大学出版社，2006.
[2] 管耀武，杨宗德.ARM嵌入式无线通信系统开发实例精讲[M].北京：电子工业出版社，2006.
[3] 周立功.ARM嵌入式系统软件开发实例(一)[M].北京：北京航空航天大学出版社，2004.

（收稿日期：2008-12-17）