第 34 卷 第 3 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.3 Computer Engineering February 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)03—0280—03 文献标识码：A 中图分类号：TN915.02

通信处理器的 Boot Loader 及 I/O 接口驱动程序

刘暾东，谢维盛，余齐齐
(厦门大学自动化系，厦门 361005)

摘 要：针对 Ethernet/IP 通信处理器的开发，采用 S3C2410 ARM9 微处理器和 Windows CE.NET 嵌入式操作系统，建立了通信处理器的 Boot Loader 和 I/O 接口驱动程序。Boot Loader 与硬件高度相关，担负着初始化系统硬件和引导操作系统的双重任务。而 I/O 接口驱动程序使操作系统自动识别外围数据采集设备，为应用程序对底层设备的操控提供服务。实验测试表明该 Ethernet/IP 通信处理器运行稳定、可靠。
关键词：Boot Loader 程序；Windows CE.NET 系统；驱动程序

Communication Processor's Boot Loader and I/O Interface Driver

LIU Tun-dong, XIE Wei-sheng, YU Qi-qi
(Automation Department, Xiamen University, Xiamen 361005)

【Abstract】Aiming at the design of the Ethernet/IP communication processor, this paper adopts S3C2410 ARM9 microprocessor and Windows CE.NET operating system to develop the communication processor's Boot Loader program and I/O interface's driver. The Boot Loader program, which is highly relevant to the hardware, initializes hardware platform and boots Windows CE.NET OS. While the I/O interface's driver makes the OS detect peripherals automatically, and provides services for the applications to control the bottom equipments. Experimental results show that the Ethernet/IP communication processor operates steadily and credibly.
【Key words】Boot Loader; Windows CE.NET; driver

Ethernet/IP 通信处理器是一种基于工业以太网协议，实现工业现场数据网络化传输的通信设备。对通信的实时性、可靠性要求很高。传统 8/16 位单片机在无法满足多任务信息处理的功能要求。针对 Ethernet/IP 通信处理器的开发，本文在硬件上采用基于 ARM 920T 内核的 S3C2410 微处理器，该处理器具有五级流水线、哈佛结构、低功耗、高性能的特点，工作频率可达 200 MHz；在软件上使用实时多任务嵌入式操作系统 Windows CE.NET，以增强系统的管理和网络通信功能，快速实现现场设备的联网功能，简化应用软件的编写。

1 通信处理器的构建

1.1 Ethernet/IP 通信处理器硬件构成

利用 S3C2410 出色的内核性能和丰富的外部接口，构造出 Ethernet/IP 通信处理器的硬件系统，其组成结构如图 1 所示。

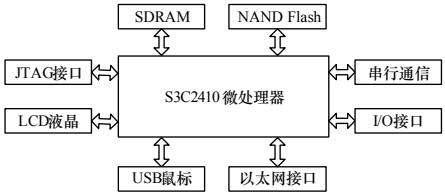

图 1 基于 S3C2410 的通信处理器硬件结构

由于 S3C2410 自身不含以太网接口，因此采用 CIRRUS 公司的 CS8900A 网卡芯片实现以太网的功能，通过正确设置网关、子网掩码等参数，可实现与局域网上终端设备的数据通信。I/O 接口的电路由地址信号、数据信号和控制信号构成，从系统总线引出 12 位地址总线、8 位数据总线和必要的控制信号，经电平转换及缓冲隔离，实现双向并行 I/O 接口功能。

1.2 Windows CE.NET 操作系统

Windows CE.NET 是微软公司推出的一款多线程多任务嵌入式操作系统。其不仅具有强大的通信功能，而且最大程度地继承了桌面版 Windows 操作系统的丰富功能，支持当前流行的软件技术和运行库，可大大缩短系统开发周期和降低技术风险[1]。目前，Windows CE.NET 广泛应用于工业控制、信息设备、移动应用、消费类电子产品等领域。

2 Boot Loader 的设计与实现

Boot Loader 是系统的引导代码，其功能与 PC 机上的 BIOS 类似。通过 Boot Loader 程序，可以初始化硬件，加载操作系统。

Boot Loader 与硬件密切相关，每个特定系统的 Boot Loader 都有所不同，因此需要自行完成 Boot Loader 程序的开发。设计一个良好的 Boot Loader 可以大大增强系统的稳定性，提高系统的实时性。

Boot Loader 由 OEM 启动代码(OEM startup code)和主代码(Main Code)两部分组成[2]。其中 OEM 启动代码是最先执行的部分，主要负责硬件平台初始化工作，之后就跳到主代码中执行；主代码主要负责下载 Windows CE 操作系统映像(OS image)，并设置合适的硬件、软件环境，以便操作系统内核的顺利启动。

2.1 启动代码的流程

启动代码是 Boot Loader 的入口点，系统上电或复位时都从地址 0x00000000 开始执行启动代码。启动代码除了完成基本软硬件环境初始化，并为启动 Windows CE 的内核的运行

基金项目：福建省科技计划基金资助项目(2004J020)
作者简介：刘暾东(1970－)，男，副教授、博士，主研方向：嵌入式智能信息处理与控制；谢维盛、余齐齐，硕士研究生
收稿日期：2007-02-09 **E-mail：**ltd@xmu.edu.cn

准备空间外，还负责为主代码的执行做准备。图 2 为 Boot Loader 的启动代码流程。

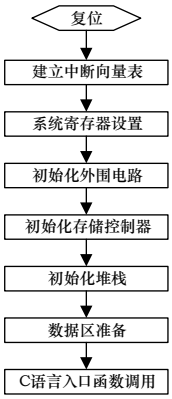

图 2　Boot Loader 启动代码的流程

2.2　主代码设计与分析

主代码是管理 Boot Loader 执行的 C 语言代码，它通过接收来自串口的用户命令或者检查相应的硬件来决定所要完成的操作。主代码主要由以下几个部分组成：以太网 I/O 代码，通过 CS8900A 网卡从远程机下载 OS Image；串口调试代码，实现在远程计算机和开发平台之间通信；Flash 写入代码，将 OS Image 写入到 Flash。

在系统启动代码执行完之后，Boot Loader 跳转到主代码的入口点 BootLoaderMain 函数，该函数控制了主代码的执行流程，主要调用了 5 个函数来完成与硬件平台相关的工作：(1)OEMDebugInit()初始化调试串口；(2)OEMPlatformInit()初始化板级硬件，如实时时钟、以太网控制器；(3)OEMPreDownload()为下载 OS Image 做准备工作，包括获取 IP、初始化 TFTP 连接；(4)DownloadImage()下载 OS Image 到目标设备的 RAM 或 Flash 中；(5)OEMLaunch()启动 OS Image。

主代码最重要的功能是将 Windows CE 的映像文件下载到目标平台上。由于 Windows CE 内核映像文件通常为 20 MB~30 MB，要将如此庞大的数据量下载到目标设备中，传输效率是关键问题。兼顾传输效率和灵活性，以太网则是十分合理的选择。远程主机与 Windows CE 目标平台的通信采用 TFTP 协议。目标平台首先通过 TFTP 发送请求报文到远程主机请求下载 Windows CE 内核的映像文件，建立 TFTP 连接并确认传输内容为 Windows CE 映像数据后，再获取映像文件的起始地址、长度等，然后开始接收 OS Image 到 RAM 中，并校验和确认数据是否有效，最后在 Flash 上分配有效地址空间将 OS Image 写入。

3　I/O 接口驱动程序的设计

通过 Boot Loader 程序，将 Windows CE.NET 移植到 Ethernet/IP 通信处理器嵌入式平台后，为了使 Windows CE 能够识别外围的数据采集设备，还需要开发 I/O 接口的驱动程序。

3.1　流接口驱动程序

目前基于 Windows CE 的驱动模型有两种：本机驱动模型和流接口驱动模型。本地驱动程序适用于集成到基于 Windows CE 的平台的设备，通常由基于 Windows CE 的原始设备制造商提供。流接口驱动程序是一个管理外围设备的动态连接库，其实现一组固定的函数称为流接口函数，这些流接口函数使得应用程序可以通过文件系统中的特殊文件与设备进行交互。流接口驱动程序几乎支持任何可以连接到基于 Windows CE 的平台的外围设备[3]。

流接口驱动程序由一个叫作设备管理器程序的特殊应用程序加载、管理和卸载。流接口驱动程序接收从设备管理器或应用程序传来的命令并实现对外围设备的控制，它的工作机制如图 3 所示。

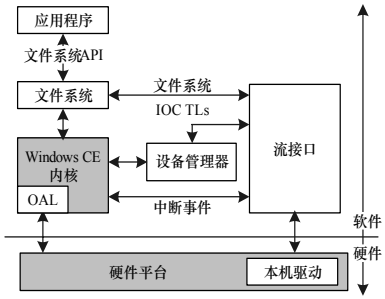

图 3　流接口驱动程序的工作机制

3.2　驱动程序的中断处理

中断是 Windows CE 驱动程序设计的一个关键，驱动程序需要实现特定设备的中断响应、中断引发的数据传送和处理。Windows CE 把中断处理分为两部分：中断服务例程(ISR)和中断服务线程(IST)。ISR 运行于内核模式，对代码的效率要求高，它的职责主要是给操作系统返回相应的逻辑中断标识，以便内核调度和启动合适的 IST。IST 在设备驱动程序软件模块中编写，执行了大多数的中断处理。当内核接到 ISR 传来的中断标识之后就会触发一个中断事件，激活正等待该事件对象的 IST，进入具体的中断处理过程。

为了使 ISR 正确返回逻辑中断标识，首先要在文件 Oalintr.h 中定义逻辑中断标识：#define SYSINTR_DIO (SYSINTR_FIRMWARE+13)；然后，修改 OEM 适配层(OAL)的中断处理函数 OEMInterruptEnable, OEMInterruptDisable, OEMInterruptDone，添加自定义中断的使能、禁止、复位代码；最后，在 OAL 中的 OEMInterruptHandler 函数中添加 ISR 程序，返回自定义的中断标识。

3.3　驱动程序的设计与实现

本文设计的 I/O 接口驱动程序采用流接口驱动模型，以中断方式处理数据。流接口驱动程序实现一组标准的流接口函数，用来完成文件 I/O 和电源管理函数，以提供给 Windows CE 内核使用。Windows CE 所要求的 10 个入口函数如表 1 所示。

表 1　流接驱动程序入口点函数描述

函数名称	描述
XXX_Init	初始化设备
XXX_Deinit	卸载设备
XXX_Open	打开设备，用于读或(和)写
XXX_Close	关闭设备
XXX_IOControl	发送命令到设备
XXX_PowerUp	给设备恢复供电
XXX_PowerDown	结束对设备的供电
XXX_Read	从设备中读取数据
XXX_Write	写数据到设备
XXX_Seek	移动设备的数据指针

表中 XXX 表示设备名前缀，在驱动中将以“DIO”代替。针对 I/O 数据扩展接口，本驱动的主要功能是设备初始化、数据读写和结束程序，因此需要实现 DIO_Init, DIO_Open, DIO_Close, DIO_Write, DIO_Read 等函数，而其他函数简单

地返回一个值即可。下面以 DIO_Init 函数为例进行分析。

DIO_Init 函数的主要任务是调用 CreateThread 函数创建一个挂起的中断服务线程 ThreadIST，并为 ThreadIST 的执行做准备工作：创建同步事件对象，设置 ThreadIST 的优先级别，注册中断等。其主要代码如下：

```
g_hevInterrupt = CreateEvent(NULL, FALSE, FALSE, NULL);
//创建一个事件对象
g_hThread = CreateThread(NULL, 0, ThreadIST, NULL,
CREATE_SUSPENDED, &dwThreadID);
//创建等待事件的 IST 线程，初始状态为挂起
CeSetThreadPriority(g_hThread, nISTPriority);
//给 IST 设置合适的优先级
InterruptInitialize(g_dwSysInt, g_hevInterrupt, NULL, 0);
//注册中断
ResumeThread( g_hThread );          //启动 ThreadIST
```

中断服务线程 ThreadIST 启动之后，需要进行虚拟内存映射和相关寄存器的配置以便最终完成中断处理。ThreadIST 的结构如下：

```
DWORD WINAPI ThreadIST(LPVOID lpParameter)
{
 DIO_InitializeAddresses();       //虚拟内存映射
 DIO_EnableInterrupt();           //相关寄存器设置
 …
 while (g_fRun)      //循环直至通知退出
 {      …
   dwStatus=WaitForSingleObject(g_hevInterrupt, INFINITE);
   //等待同步事件
       if (dwStatus == WAIT_OBJECT_0)
       {      //此处进行中断处理
              …
              //通知内核中断处理已完成
              InterruptDone( g_dwSysInt );
       }
 }
…
}
```

DIO_InitializeAddresses()和 DIO_EnableInterrupt()是用户自定义的函数。分别用于完成虚拟内存映射和相关寄存器的配置。本驱动在用户模式下运行，需要用虚拟内存映射的方法，实现对 I/O 接口物理空间的间接访问。DIO_EnableInterrupt()用来将 S3C2410X 的管脚 GPF2 配置为外部中断 EINT2，并设定触发方式为下降沿。

ThreadIST 在完成初始化工作后进入循环，首先调用 WaitForSingleObject 函数阻塞线程，等待中断事件，直到触发事件从内核返回。一旦产生中断，IST 将与 I/O 接口通信，并从 I/O 接口中读取所有必要的数据，完成其中断交互操作。然后，IST 用关联的逻辑中断标识来调用 InterruptDone 函数，通知内核中断处理已完成。内核将重新启用指定的中断，以便接收该设备的下一个中断请求。

3.4 驱动程序的封装

在完成驱动程序的编写并导出流接口函数之后，还需将驱动程序封装进操作系统。为此，首先在注册表文件 Platform.reg 中添加 DIO 子键和相应键值，具体如下：

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\DIOPort]
    "Index"=dword:1        ; 设备文件名索引
    "Prefix"="DIO"         ; 设备文件的前缀
    "Dll"="DIOPort.dll"    ; 设备驱动 DLL 文件名
    "Order"=dword:0        ; 驱动程序的加载顺序
```

然后在 Platform.bib 文件的 MODULE 部分添加如下代码，使驱动程序文件被包含在最终生成的映像文件中：

```
DIOPort.dll $(_FLATRELEASEDIR)\DIOPort.dll NK SH
```

4 程序测试

通过 CS8900A 网卡将 OS Image 下载到 Ethernet/IP 通信处理器的 Flash 中，实际测试速度可以达到 200 Kb/s 以上，表明 Boot Loader 很好地完成了下载 OS Image 的任务。

另外，I/O 接口驱动程序没有界面，完全在系统底层运行，为观察驱动程序的运行状态，编写了 I/O 驱动测试程序，借助 LCD 对数据进行观察。试验结果表明，该驱动程序能准确、快速、稳定地实现数据的读取与写入。

5 结束语

S3C2410 芯片与 Windows CE.NET 的结合应用满足了 Ethernet/IP 通信处理器在硬件和软件资源上的需求。针对 S3C2410 与 Windows CE.NET 的 Boot Loader 和 I/O 接口驱动程序的开发是该通信处理器开发中关键的一步，为系统应用软件开发奠定了基础。

参考文献

[1] 周毓林, 宁 杨, 陆贵强, 等. Windows CE.net 内核定制及开发应用[M]. 北京: 电子工业出版社, 2005.

[2] 微软公司. How to Develop a Boot Loader[EB/OL]. [2006-11-20]. http://msdn2.microsoft.com/en-us/library/ms903967.aspx.

[3] 微软公司. Windows CE 设备驱动程序开发指南[M]. 希望图书创作室, 译. 北京: 希望电子出版社, 1999.

（上接第 273 页）

4 结束语

控制器作为分布式实时控制系统的重要应用领域，也存在着平台多样性和系统异构性的问题，开放性和分布式正成为其发展的主要方向。利用中间件构筑开放式控制器体系，可以降低软件开发的费用，增强应用的互操作性、可重用性、可扩展性和可维护性。同时由于中间件技术的日趋完善，它为控制器的体系结构注入了新鲜的活力。

参考文献

[1] Schmidt D C, Levine D L, Mungee S. The Design of the TAO Real-time Object Request Broker[J]. Computer Communications, 1998, 21(4): 294-324.

[2] Object Management Group. Real-time CORBA Specification (Version1.1)[EB/OL]. (2002-08-02). http://www. omg. Org.

[3] 潘慧芳, 周兴社. CORBA 构件模型综述[J]. 计算机应用研究, 2005, 22(5): 14-16.

[4] 迟永琳, 王宇晗, 吴祖育, 等. 开放结构控制器构件化实现方法[J]. 机床与液压, 2003, (1): 138-140.

[5] 张 磊, 樊留群. 基于 CORBA 技术的矿山机械远程监控系统[J]. 计算机工程, 2006, 32(12): 250-252.