“黑色经典”系列之《嵌入式 Linux 应用程序开发详解》

第 4 章 嵌入式系统基础

本章目标

从本章开始，读者开始真正进入嵌入式领域学习。本章讲解嵌入式系统的基础知识及基本服务的配置，学习完本章读者将掌握如下内容。

- 了解嵌入式系统的含义及其发展情况
- 了解嵌入式系统的体系结构
- 了解 ARM 处理器及 ARM9 的相关知识
- 熟悉三星处理器 S3C2410
- 了解嵌入式系统的基本调试手段

4.1 嵌入式系统概述

4.1.1 嵌入式系统简介

尼葛洛庞帝 2001 年访华时的预言“4～5 年后，嵌入式智能电脑将是继 PC 和 Internet 后的最伟大发明！”如今，嵌入式系统已成功当今最为热门的领域之一，它迅猛的发展势头引起了社会各方面人士的关注。如家用电器、手持通信设备、信息终端、仪器仪表、汽车、航天航空、军事装备、制造工业、过程控制等。今天，嵌入式系统带来的工业年产值已超过 1 万亿美元。用市场观点来看，PC 已经从高速增长进入到平稳发展时期，其年增长率由 20 世纪 90 年代中期的 35%逐年下降，使单纯由 PC 机带领电子产业蒸蒸日上的时代成为历史。根据 PC 时代的概念，美国 *Business Week* 杂志提出了“后 PC 时代”概念，即计算机、通信和消费产品的技术将结合起来，以 3C 产品的形式通过 Internet 进入家庭。这必将培育出一个庞大的嵌入式应用市场。那么究竟什么是嵌入式系统呢？

按照电器工程协会的定义，嵌入式系统是用来控制或者监视机器、装置、工厂等大规模系统的设备。这个定义主要是从嵌入式系统的用途方面来进行定义的。

那么，下面再来看一个在多数书籍资料中的关于嵌入式系统的定义：嵌入式系统是指以应用为中心，以计算机技术为基础，软件硬件可剪裁，适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。它主要由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户应用软件等部分组成。它具有“嵌入性”、“专用性”与“计算机系统”的三个基本要素。

从这个定义可以看出，人们平常所广泛使用的手机、PDA、MP3、机顶盒都属于嵌入式系统设备；而车载 GPS 系统、机器人也是属于嵌入式系统。图 4.1 展出了人们日常生活中形形色色的嵌入式产品。的确，嵌入式系统已经进入了人们生活的方方面面。

图 4.1　生活中的嵌入式设备

4.1.2　嵌入式系统发展历史

嵌入式系统经过 30 年的发展历程，主要经历了 4 个阶段。

第 1 阶段是以单芯片为核心的可编程控制器形式的系统。这类系统大部分应用于一些专业性强的工业控制系统中，一般没有操作系统的支持，通过汇编语言编程对系统进行直接控制。这一阶段系统的主要特点是：系统结构和功能相对单一，处理效率较低，存储容量较小，几乎没有用户接口。由于这种嵌入式系统使用简单、价格低，因此以前在国内工业领域应用较为普遍，但是现在已经远不能适应高效的、需要大容量存储的现代工业控制和新兴信息家电等领域的需求。

第 2 阶段是以嵌入式 CPU 为基础、以简单操作系统为核心的嵌入式系统。其主要特点是：CPU 种类繁多，通用性比较弱；系统开销小，效率高；操作系统达到一定的兼容性和扩展性；应用软件较专业化，用户界面不够友好。

第 3 阶段是以嵌入式操作系统为标志的嵌入式系统。其主要特点是：嵌入式操作系统能运行于各种不同类型的微处理器上，兼容性好；操作系统内核小、效率高，并且具有高度的模块化和扩展性；具备文件和目录管理、支持多任务、支持网络应用、具备图形窗口和用户界面；具有大量的应用程序接口 API，开发应用程序较简单；嵌入式应用软件丰富。

第 4 阶段是以 Internet 为标志的嵌入式系统。这是一个正在迅速发展的阶段。目前大多数嵌入式系统还孤立于 Internet 之外，但随着 Internet 的发展以及 Internet 技术与信息家电、

工业控制技术结合日益密切，嵌入式设备与 Internet 的结合将代表嵌入式系统的未来。

4.1.3 嵌入式系统的特点

（1）面向特定应用的特点。从前面图 1.1 中也可以看出，嵌入式系统与通用型系统的最大区别就在于嵌入式系统大多工作在为特定用户群设计的系统中，因此它通常都具有低功耗、体积小、集成度高等特点，并且可以满足不用应用的特定需求。

（2）嵌入式系统的硬件和软件都必须进行高效地设计，量体裁衣、去除冗余，力争在同样的硅片面积上实现更高的性能，这样才能在具体应用中对处理器的选择更具有竞争力。

（3）嵌入式系统是将先进的计算机技术、半导体技术和电子技术与各个行业的具体应用相结合后的产物。这一点就决定了它必然是一个技术密集、资金密集、高度分散、不断创新的知识集成系统，从事嵌入式系统开发的人才也必须是复合型人才。

（4）为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存储于磁盘中。

（5）嵌入式开发的软件代码尤其要求高质量、高可靠性，由于嵌入式设备所处的环境往往是无人职守或条件恶劣的情况下，因此，其代码必须有更高的要求。

（6）嵌入式系统本身不具备二次开发能力，即设计完成后用户通常不能对其中的程序功能进行修改，必须有一套开发工具和环境才能进行再次开发。

4.1.4 嵌入式系统的体系结构

嵌入式系统作为一类特殊的计算机系统，一般包括以下 3 个方面：硬件设备、嵌入式操作系统和应用软件。它们之间的关系如图 4.2 所示。

硬件设备包括嵌入式处理器和外围设备。其中的嵌入式处理器（CPU）是嵌入式系统的核心部分，它与通用处理器最大的区别在于，嵌入式处理器大多工作在为特定用户群所专门设计的系统中，它将通用处理器中许多由板卡完成的任务集成到芯片内部，从而有利于嵌入式系统在设计时趋于小型化，同时还具有很高的效率和可靠性。如今，全世界嵌入式处理器已经超过 1000 多种，流行的体系结构有 30 多个系列，其中以 ARM、PowerPC、MC 68000、MIPS 等使用得最为广泛。

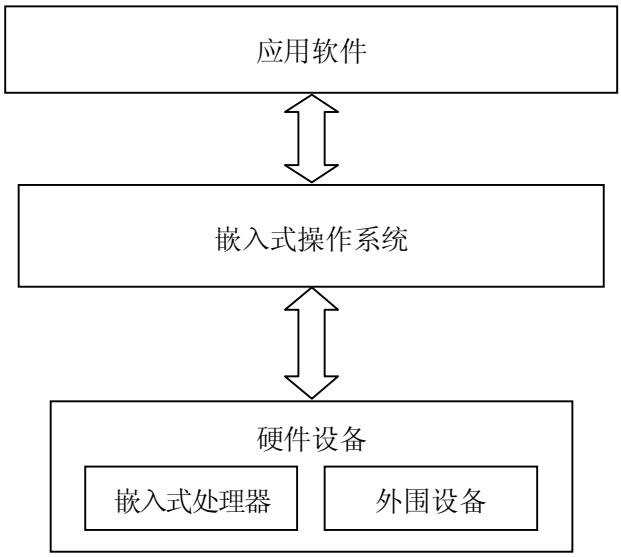

图 4.2　嵌入式体系结构图

外围设备是嵌入式系统中用于完成存储、通信、调试、显示等辅助功能的其他部件。目前常用的嵌入式外围设备按功能可以分为存储设备（如 RAM、SRAM、Flash 等）、通信设备（如 RS-232 接口、SPI 接口、以太网接口等）和显示设备（如显示屏等）3 类。

常见存储器概念辨析：RAM、SRAM、SDRAM、ROM、EPROM、EEPROM、Flash

存储器可以分为很多种类，其中根据掉电数据是否丢失可以分为 RAM（随机存取存储器）和 ROM（只读存储器），其中 RAM 的访问速度比较快，但掉电后数据会丢失，而 ROM 掉电后数据不会丢失。人们通常所说的内存即指系统中的 RAM。

RAM 又可分为 SRAM（静态存储器）和 DRAM（动态存储器）。SRAM 是利用双稳态触发器来保存信息的，只要不掉电，信息是不会丢失的。DRAM 是利用 MOS（金属氧化物半导体）电容存储电荷来储存信息，因此必须通过不停的给电容充电来维持信息，所以 DRAM 的成本、集成度、功耗等明显优于 SRAM。

而通常人们所说的 SDRAM 是 DRAM 的一种，它是同步动态存储器，利用一个单一的系统时钟同步所有的地址数据和控制信号。使用 SDRAM 不但能提高系统表现，还能简化设计、提供高速的数据传输。在嵌入式系统中经常使用。

EPROM、EEPROM 都是 ROM 的一种，分别为可擦除可编程 ROM 和电可擦除 ROM，但使用不是很方便。

Flash 也是一种非易失性存储器（掉电不会丢失），它擦写方便，访问速度快，已大大取代了传统的 EPROM 的地位。由于它具有和 ROM 一样掉电不会丢失的特性，因此很多人称其为 Flash ROM。

嵌入式操作系统从嵌入式发展的第 3 阶段起开始引入。嵌入式操作系统不仅具有通用操作系统的一般功能，如向上提供对用户的接口（如图形界面、库函数 API 等），向下提供与硬件设备交互的接口（硬件驱动程序等），管理复杂的系统资源，同时，它还在系统实时性、

硬件依赖性、软件固化性以及应用专用性等方面，具有更加鲜明的特点。

应用软件是针对特定应用领域，基于某一固定的硬件平台，用来达到用户预期目标的计算机软件。由于嵌入式系统自身的特点，决定了嵌入式应用软件不仅要求做到准确性、安全性和稳定性等方面需要，而且还要尽可能地进行代码优化，以减少对系统资源的消耗，降低硬件成本。

4.1.5 几种主流嵌入式操作系统分析

1．嵌入式 Linux

嵌入式 Linux（Embedded Linux）是指对标准 Linux 经过小型化裁剪处理之后，能够固化在容量只有几 KB 或者几 MB 字节的存储器芯片或者单片机中，是适合于特定嵌入式应用场合的专用 Linux 操作系统。在目前已经开发成功的嵌入式系统中，大约有一半使用的是 Linux。这与它自身的优良特性是分不开的。

嵌入式 Linux 同 Linux 一样，具有低成本、多种硬件平台支持、优异的性能和良好的网络支持等优点。另外，为了更好地适应嵌入式领域的开发，嵌入式 Linux 还在 Linux 基础上做了部分改进，如下所示。

- 改善的内核结构

Linux 内核采用的是整体式结构（Monolithic），整个内核是一个单独的、非常大的程序，这样虽然能够使系统的各个部分直接沟通，提高系统响应速度，但与嵌入式系统存储容量小、资源有限的特点不相符合。因此，在嵌入式系统经常采用的是另一种称为微内核（Microkernel）的体系结构，即内核本身只提供一些最基本的操作系统功能，如任务调度、内存管理、中断处理等，而类似于文件系统和网络协议等附加功能则运行在用户空间中，并且可以根据实际需要进行取舍。这样就大大减小了内核的体积，便于维护和移植。

- 提高的系统实时性

由于现有的 Linux 是一个通用的操作系统，虽然它也采用了许多技术来加快系统的运行和响应速度，但从本质上来说并不是一个嵌入式实时操作系统。因此，利用 Linux 作为底层操作系统，在其上进行实时化改造，从而构建出一个具有实时处理能力的嵌入式系统，如 RT-Linux 已经成功地应用于航天飞机的空间数据采集、科学仪器测控和电影特技图像处理等各种领域。

嵌入式 Linux 同 Linux 一样，也有众多的版本，其中不同的版本分别针对不同的需要在内核等方面加入了特定的机制。嵌入式 Linux 的主要版本如表 4.1 所示。

表 4.1 嵌入式 Linux 主要版本

版　本	简 单 介 绍
μCLinux	开放源码的嵌入式 Linux 的典范之作。它主要是针对目标处理器没有存储管理单元 MMU，其运行稳定，具有良好的移植性和优秀的网络功能，对各种文件系统有完备的支持，并提供标准丰富的 API
RT-Linux	由美国墨西哥理工学院开发的嵌入式 Linux 硬实时操作系统。它已有广泛的应用
Embedix	根据嵌入式应用系统的特点重新设计的 Linux 发行版本。它提供了超过 25 种的 Linux

	系统服务，包括 Web 服务器等。此外还推出了 Embedix 的开发调试工具包、基于图形界面的浏览器等。可以说，Embedix 是一种完整的嵌入式 Linux 解决方案
XLinux	采用了“超字元集”专利技术，使 Linux 内核不仅能与标准字符集相容，还涵盖了 12 个国家和地区的字符集。因此，XLinux 在推广 Linux 的国际应用方面有独特的优势
PoketLinux	它可以提供跨操作系统并且构造统一的、标准化的和开放的信息通信基础结构，在此结构上实现端到端方案的完整平台
红旗嵌入式 Linux	由北京中科院红旗软件公司推出的嵌入式 Linux，它是国内做得较好的一款嵌入式操作系统。目前，中科院计算机研究所自行开发的开放源码的嵌入式操作系统——Easy Embedded OS（EEOS）也已经开始进入实用阶段了

为了不失一般性，本书说所用的嵌入式 Linux 是标准内核裁减的 Linux，而不是上表中的任何一种。

2．VxWorks

VxWorks 操作系统是美国 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统（RTOS），它是在当前市场占有率最高的嵌入式操作系统。VxWorks 的实时性做得非常好，其系统本身的开销很小，进程调度、进程间通信、中断处理等系统公用程序精练而有效，使得它们造成的延迟很短。另外 VxWorks 提供的多任务机制，对任务的控制采用了优先级抢占（Linux 2.6 内核也采用了优先级抢占的机制）和轮转调度机制，这充分保证了可靠的实时性，并使同样的硬件配置能满足更强的实时性要求。另外 VxWorks 具有高度的可靠性，从而保证了用户工作环境的稳定。同时，VxWorks 还有很完备强大的集成开发环境，这也大大方便了用户的使用。

但是，由于 VxWorks 的开发和使用都需要交高额的专利费，因此大大增加了用户的开发成本。同时，由于 VxWorks 的源码不公开，造成它部分功能的更新（如网络功能模块）滞后。

3．QNX

QNX 是业界公认的 X86 平台上最好的嵌入式实时操作系统之一，它具有独一无二的微内核实时平台，是建立在微内核和完全地址空间保护基础之上的，它同样具有实时性强、稳定可靠的优点。

4．Windows CE

WINDOWS CE 是微软开发的一个开放的、可升级的 32 位嵌入式操作系统，是基于掌上型电脑类的电子设备操作系统。它是精简的 Windows 95。Windows CE 的图形用户界面相当出色。Windows CE 具有模块化、结构化和基于 Win32 应用程序接口以及与处理器无关等特点。它不仅继承了传统的 Windows 图形界面，并且用户在 Windows CE 平台上可以使用 Windows 95/98 上的编程工具（如 Visual Basic、Visual++等）、也可以使用同样的函数、使用同样的界面风格，使绝大多数 Windows 上的应用软件只需简单的修改和移植就可以在 WindowsCE 平台上继续使用。但与 VxWorks 相同，WindowsCE 也是比较昂

贵的。

5．Palm OS

Paml OS 在 PDA 和掌上电脑有着很大的用户群。Plam OS 最明显的特点在精简，它的内核只有几千个字节，同时用户也可以方便地开发定制，具有较强的可操作性。

4.2 ARM 处理器硬件开发平台

4.2.1 ARM 处理器简介

ARM 是一类嵌入式微处理器，同时也是一个公司的名字。ARM 公司于 1990 年 11 月成立于英国剑桥，它是一家专门从事 16/32 位 RISC 微处理器知识产权设计的供应商。ARM 公司本身不直接从事芯片生产，而只是授权 ARM 内核，再给生产和销售半导体的合作伙伴，同时也提供基于 ARM 架构的开发设计技术。世界各大半导体生产商从 ARM 公司处购买其设计的 ARM 微处理器核，根据各自不同的应用领域，加入适当的外围电路，从而形成自己的 ARM 微处理器芯片进入市场。

ARM 公司从成立至今，在短短几十年的时间就占据了 75%的市场份额，如今，ARM 微处理器及技术的应用几乎已经深入到各个领域。采用 ARM 技术的微处理器现在已经遍及各类电子产品，汽车、消费娱乐、影像、工业控制、海量存储、网络、安保和无线等市场。到 2001 年就几乎已经垄断了全球 RISC 芯片市场，成为业界实际的 RISC 芯片标准。图 4.3 列举了使用 ARM 微处理器的公司名称。

图 4.3 采用 ARM 微处理器的公司

ARM 的成功，一方面得益于它独特的公司运作模式，另一方面，当然来自于 ARM 处理器自身的优良性能。ARM 处理器有如下特点。

- 体积小、低功耗、低成本、高性能。
- 支持 Thumb（16 位）/ARM（32 位）双指令集，能很好的兼容 8 位/16 位器件。
- 大量使用寄存器，指令执行速度更快。
- 大多数数据操作都在寄存器中完成。
- 寻址方式灵活简单，执行效率高。
- 指令长度固定。

小知识

常见的 CPU 指令集分为 CISC 和 RISC 两种。

CISC（Complex Instruction Set Computer）是“复杂指令集”。自 PC 机诞生以来，32 位以前的处理器都采用 CISC 指令集方式。由于这种指令系统的指令不等长，因此指令的数目非常多，编程和设计处理器时都较为麻烦。但由于基于 CISC 指令架构系统设计的软件已经非常普遍了，所以包括 Intel、AMD 等众多厂商至今使用的仍为 CISC。

RISC（Reduced Instruction Set Computing）是“精简指令集”。研究人员在对 CISC 指令集进行测试时发现，各种指令的使用频度相当悬殊，其中最常使用的是一些比较简单的指令，它们仅占指令总数的 20%，但在程序中出现的频度却占 80%。RISC 正是基于这种思想提出的。采用 RISC 指令集的微处理器处理能力强，并且还通过采用超标量和超流水线结构，大大增强并行处理能力。

4.2.2 ARM 体系结构简介

1．ARM 微处理器工作状态

ARM 微处理器的工作状态一般有两种，并可在两种状态之间切换。

- 第一种为 ARM 状态，此时处理器执行 32 位的字对齐的 ARM 指令。
- 第二种为 Thumb 状态，此时处理器执行 16 位的、半字对齐的 Thumb 指令。

2．ARM 体系结构的存储格式

- 大端格式：在这种格式中，字数据的高字节存储在低地址中，而字数据的低字节则存放在高地址中。
- 小端格式：与大端存储格式相反，在小端存储格式中，低地址中存放的是字数据的低字节，高地址存放的是字数据的高字节。

3．ARM 处理器模式

ARM 微处理器支持 7 种运行模式，分别如下。

- 用户模式（usr）：ARM 处理器正常的程序执行状态。
- 快速中断模式（fiq）：用于高速数据传输或通道处理。
- 外部中断模式（irq）：用于通用的中断处理。
- 管理模式（svc）：操作系统使用的保护模式。

- 数据访问终止模式（abt）：当数据或指令预取终止时进入该模式，可用于虚拟存储及存储保护。
- 系统模式（sys）：运行具有特权的操作系统任务。

4.2.3 ARM9 体系结构

1．ARM 微处理器系列简介

ARM 微处理器系列主要特点及其主要应用领域如表 4.2 所示。

表 4.2 ARM 微处理器系列

型　　号	主 要 特 点
ARM7	低功耗的 32 位 RISC 处理器，最适合用于对价位和功耗要求较高的消费类应用。 主要应用领域为：工业控制、Internet 设备、网络和调制解调器设备、移动电话等多种多媒体和嵌入式应用
ARM9	在高性能和低功耗特性方面提供最佳的性能。 主要应用领域为：无线设备、仪器仪表、安全系统、机顶盒、高端打印机、数字照相机和数字摄像机等
ARM9E	综合处理器，使用单一的处理器内核，提供了微控制器、DSP、Java 应用系统的解决方案，极大的减少了芯片的面积和系统的复杂程度。

续表

型　　号	主 要 特 点
ARM9E	主要应用领域为：下一代无线设备、数字消费品、成像设备、工业控制、存储设备和网络设备等
ARM10E	高性能、低功耗。由于采用了新的体系结构，与同等的 ARM9 器件相比较，在同样的时钟频率下，性能提高近 50%，同时，ARM10E 系列微处理器采用了两种先进的节能方式，使其功耗极低。 主要应用领域为：下一代无线设备、数字消费品、成像设备、工业控制、存储设备和网络设备等
SecurCore	专为安全需要而设计，带有灵活的保护单元，以确保操作系统和应用数据的安全。 主要应用领域为：对安全性要求较高的应用产品及应用系统，如电子商务、电子政务、电子银行业务、网络和认证系统等领域
StrongARM	融合了 Intel 公司的设计和处理技术以及 ARM 体系结构的电源效率，采用在软件上兼容 ARMv4 体系结构、同时采用具有 Intel 技术优点的体系结构。 主要应用领域为：便携式通讯产品和消费类电子产品

2．ARM9 主要特点

ARM 处理器凭借它的低功耗、高性能等特点，被广泛应用于个人通信等嵌入式领域，而 ARM7 也曾在中低端手持设备中占据了一席之地。然而，ARM7 的处理性能逐渐无法满足人们日益增长的高性能功能需求的处理，它开始退出主流应用领域，取而代之的是性能更加强大的 ARM9 系列处理器。

新一代的 ARM9 处理器，通过全新的设计，能够达到两倍以上于 ARM7 处理器的处理能力。它的主要特点如下所述。

（1）5 级流水线

ARM7 处理器采用的 3 级流水线设计，而 ARM9 则采用 5 级流水线设计，如图 4.4 所示。

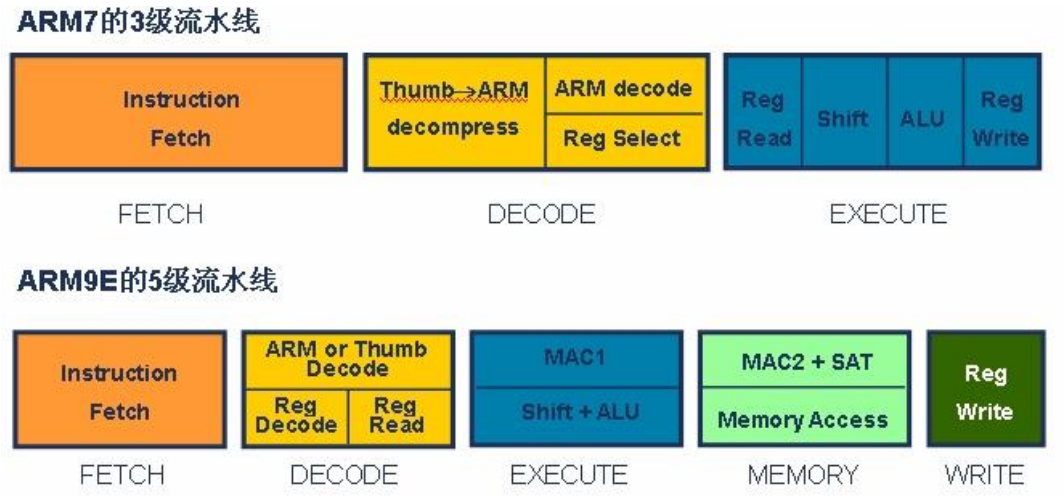

图 4.4　ARM7 与 ARM9 流水线比较

通过使用 5 级流水线机制，在每一个时钟周期内可以同时执行 5 条指令。这样就大大提高了处理性能。在同样的加工工艺下，ARM9 处理器的时钟频率是 ARM7 的 1.8～2.2 倍。

（2）采用哈佛结构

首先读者需要了解什么叫哈佛结构？在计算机中，根据计算机的存储器结构及其总线连接形式，计算机系统可以被分为冯・诺依曼结构和哈佛结构，其中冯・诺依曼结构共用数据存储空间和程序存储空间，它们共享存储器总线，这也是以往设计时常用的方式；而哈佛结构则具有分离的数据和程序空间及分离的访问总线。所以哈佛结构在指令执行时，取址和取数可以并行，因此具有更高的执行效率。ARM9 采用的就是哈佛结构，而 ARM7 采用的则是冯・诺依曼结构。如图 4.5 和图 4.6 分别体现了冯・诺依曼结构和哈佛结构的数据存储方式。

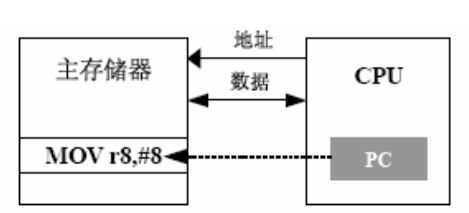

图 4.5　冯・诺依曼结构

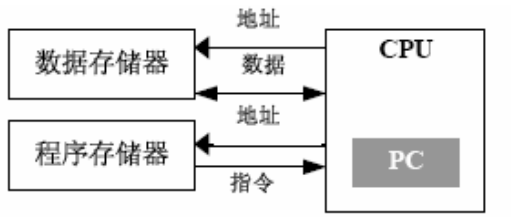

图 4.6　哈佛结构

由于在 RISC 架构的处理器中，程序中大约有 30%的指令是 Load-Store 指令，而采用哈佛结构大大提升了这两个指令的执行速度，因此对提高系统效率的贡献是非常明显的。

（3）高速缓存和写缓存的引入

由于在处理器中，一般处理器速度远远高于存储器访问速度，那么，如果存储器访问成

为系统性能的瓶颈，则处理器再快也都毫无作用。在这种情况下，高速缓存（Cache）和写缓存（Write Buffer）可以很好地解决这个问题，它们存储了最近常用的代码和数据，以供 CPU 快速存储，如图 4.7 所示：

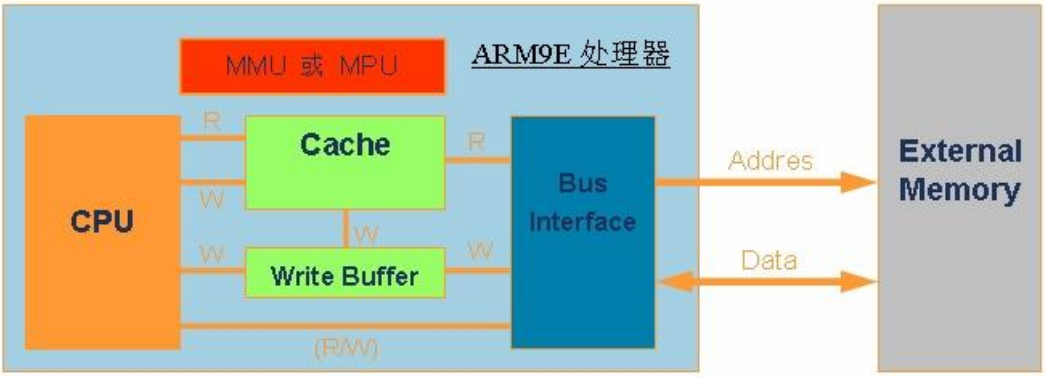

图 4.7　ARM9 的高速缓存和读缓存

（4）支持 MMU

MMU 是内存管理单元，它把内存以“页（page）”为单位来进行处理。一页内存是指一个具有一定大小的连续的内存块，通常为 4096B 或 8192B。操作系统为每个正在运行的程序建立并维护一张被称为进程内存映射（Process Memory Map）的表，表中记录了程序可以存取的所有内存页以及它们的实际位置。

每当程序存取一块内存时，它会把相应的虚拟地址（virtual address）传送给 MMU，而 MMU 会在 PMM 中查找这块内存的实际位置，也就是物理地址（physical address），物理地址可以在内存中或磁盘上的任何位置。如果程序要存取的位置在磁盘上，就必须把包含该地址的页从磁盘上读到内存中，并且必须更新 PMM 以反映这个变化（这被称为 pagefault，即页错）。MMU 的实现过程如图 4.8 所示。

只有拥有了 MMU 才能真正实现内存保护。例如当 A 进程的程序试图直接访问属于 B 进程的虚拟地址中的数据，那么 MMU 会产生一个异常（Exception）来阻止 A 的越界操作。这样，通过内存保护，一个进程的失败并不会影响其他进程的运行，从而增强了系统的稳定性，如图 4.9 所示。ARM9 也正是因此拥有了 MMU，比 ARM7 有了更强的稳定性和可靠性。

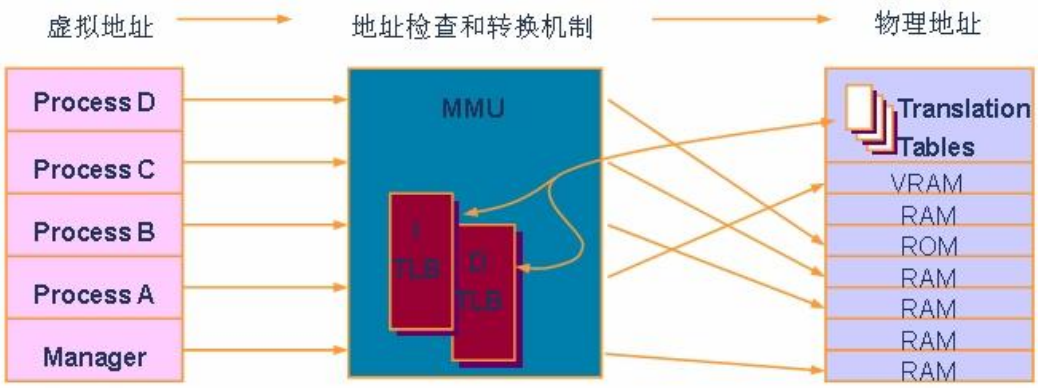

图 4.8　MMU 的实现过程

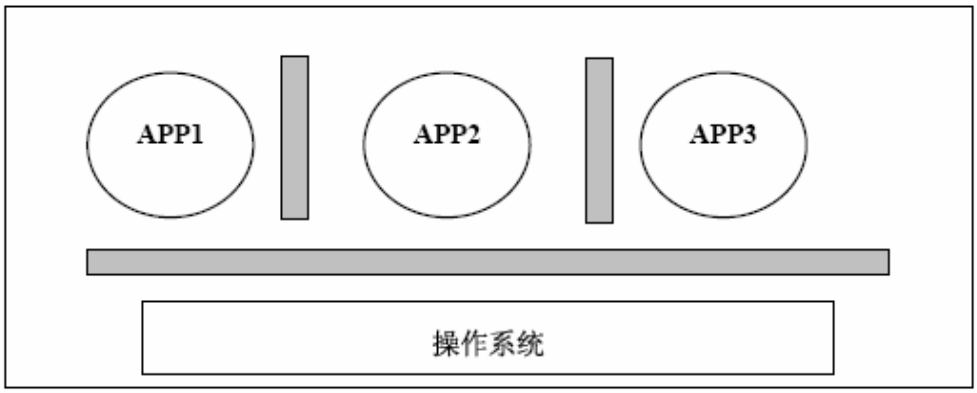

图 4.9　内存保护示意图

4.2.4　S3C2410 处理器详解

本书所采用的硬件平台是深圳优龙科技有限公司的开发板 FS2410（如图 4.10 所示），它的中央处理器是三星公司的 S3C2410X。S3C2410X 是使用 ARM920T 核、采用 0.18um 工艺 CMOS 标准宏单元和存储编译器开发而成的。由于采用了由 ARM 公司设计的 16/32 位 ARM920T RISC 处理器，因此 S3C2410X 实现了 MMU 和独立的 16KB 指令和 16KB 数据哈佛结构的缓存，且每个缓存均为 8 个字长度的流水线。它的低功耗、精简而出色的全静态设计特别适用于对成本和功耗敏感的领域。

S3C2410X 提供全面的、通用的片上外设，大大降低系统的成本，下面列举了 S3C2410X 的主要片上功能。

- 1.8V ARM920T 内核供电，1.8V/2.5V/3.3V 存储器供电；
- 16KB 指令和 16KB 数据缓存的 MMU 内存管理单元；
- 外部存储器控制（SDRAM 控制和芯片选择逻辑）；
- 提供 LCD 控制器（最大支持 4K 色的 STN 或 256K 色 TFT 的 LCD），并带有 1 个通道的 LCD 专用 DMA 控制器；
- 提供 4 通道 DMA，具有外部请求引脚；

- 提供 3 通道 UART（支持 IrDA1.0，16 字节发送 FIFO 及 16 字节接收 FIFO）/2 通道 SPI 接口；
- 提供 1 个通道多主 IIC 总线控制器/1 通道 IIS 总线控制器；
- 兼容 SD 主机接口 1.0 版及 MMC 卡协议 2.11 版；
- 提供 2 个主机接口的 USB 口/1 个设备 USB 口（1.1 版本）；
- 4 通道 PWM 定时器/1 通道内部计时器；

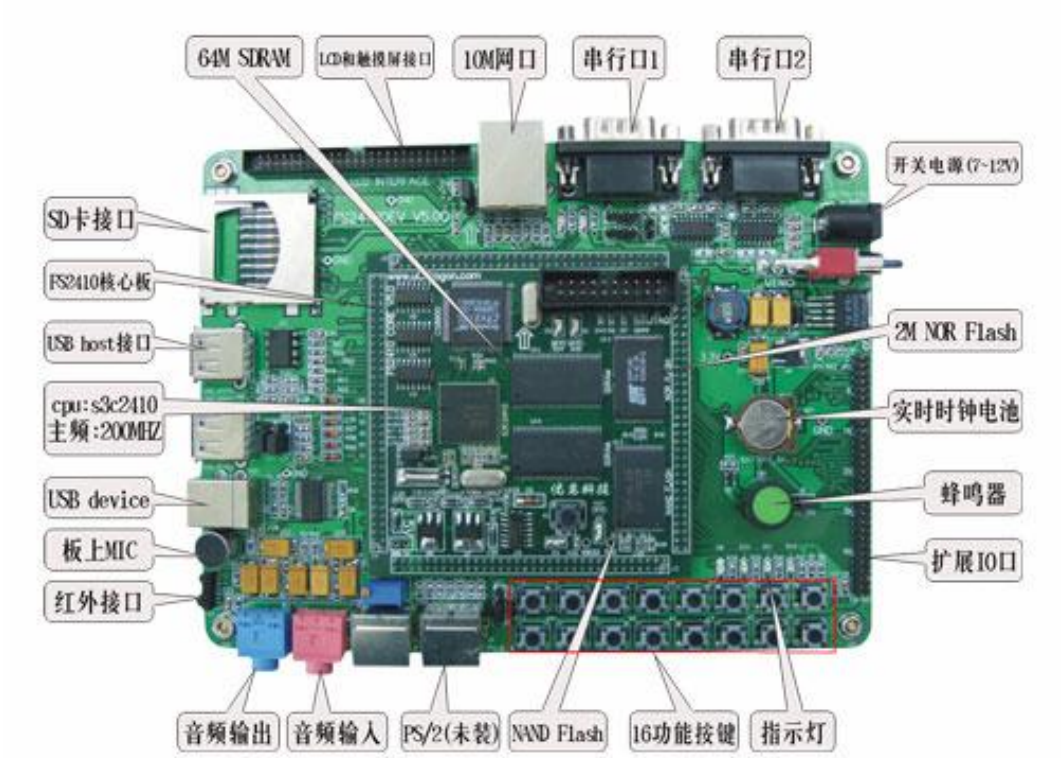

图 4.10　优龙 FS2410 开发板实物图

- 提供看门狗定时器；
- 提供 117 个通用 I/O 口/24 通道外部中断源；
- 提供电源控制不同模式：正常、慢速、空闲及电源关闭模式；
- 提供带触摸屏接口的 8 通道 10 位 ADC；
- 提供带日历功能的实时时钟控制器（RTC）；
- 具有 PLL 的片上时钟发生器。

S3C2410X 系统结构图如图 4.11 所示。

下面依次对 S3C2410X 的系统管理器、NAND Flash 引导装载器、缓冲存储器、时钟和电源管理及中断控制进行简要讲解，要注意，其中所有模式的选择都是通过相关寄存器特定值的设定来实现的，因此，当读者需要对此进行修改时，请参阅三星公司提供 S3C2410X 用户手册。

1．系统管理器

S3C2410X 支持小/大端模式，它将系统的存储空间分为 8 个组（bank），其中每个 bank 有 128MB，总共为 1GB。每个组可编程的数据总线宽度为 8/16/32 位，其中 bank0~bank5 具有固定的 bank 起始地址和结束地址，用于 ROM 和 SRAM。而 bank6 和 bank7 是大小可变的，用于 ROM、SRAM 或 SDRAM。这里，所有的存储器 bank 都具有可编程的操作周期，并且支持掉电时的 SDRAM 自刷新模式和多种类型的引导 ROM。

2．NAND Flash 引导装载器

S3C2410X 支持从 NAND flash 存储器启动，其中，开始的 4KB 为内置缓冲存储器，它

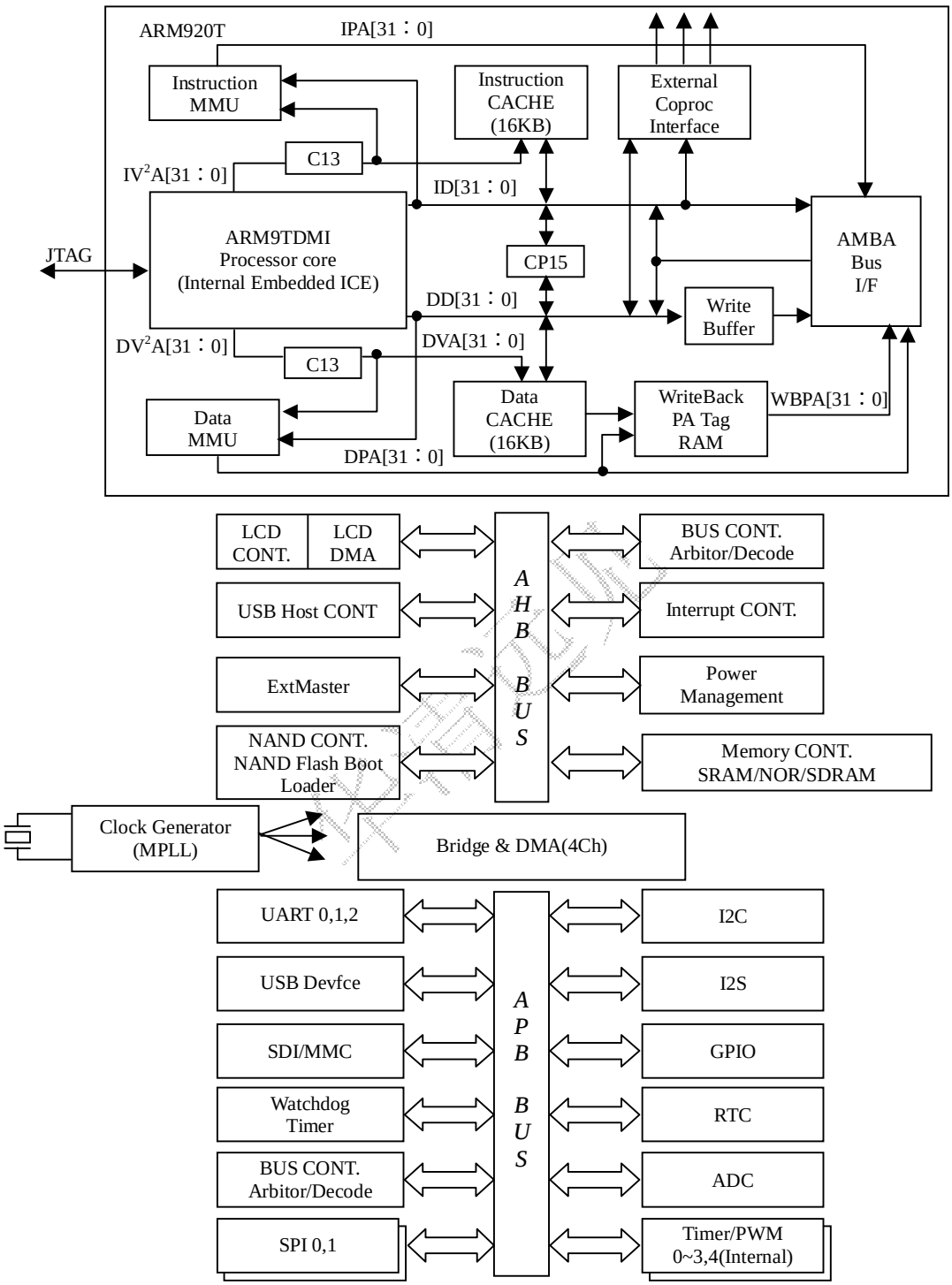

图 4.11　S3C2410X 系统结构图

在启动时将被转载（装在 or 转载）到 SDRAM 中并执行引导，之后该 4KB 可以用作其他

用途。

小知识

Flash 是一种非易失闪存技术。Intel 于 1988 年首先开发出 NOR Flash 技术之后，彻底改变了原先由 EPROM 和 EEPROM 一统天下的局面。紧接着，1989 年东芝公司发布了 NAND Flash 结构，强调降低每比特的成本、更高的性能，并且像磁盘一样可以通过接口轻松升级。

NOR Flash 的特点是芯片内执行（Execute In Place），这样应用程序可以直接在 flash 闪存内运行，而不必再把代码读到系统 RAM 中。NOR Flash 的传输效率很高，在 1～4MB 的小容量时具有很高的成本效益，但是很低的写入和擦除速度大大影响了它的性能。

NAND Flash 结构能提供极高的单元密度，可以达到高存储密度，NAND 读和写操作采用 512 字节的块，单元尺寸几乎是 NOR 器件的一半，同时由于生产过程更为简单，大大降低了生产的成本。NAND 闪存中每个块的最大擦写次数是 100 万次，是 NOR Flash 的 10 倍，这些都使得 NAND Flash 越来越受到人们的欢迎。

同时，S3C2410X 也支持从外部 nGCS0 片选的 NorFlash 启动，如在优龙的开发板上将 JP1 跳线去掉就可从 NorFlash 启动（默认从 NAND Flash 启动）。在这两种启动模式下，各片选的存储空间分配是不同的，如图 4.12 所示。

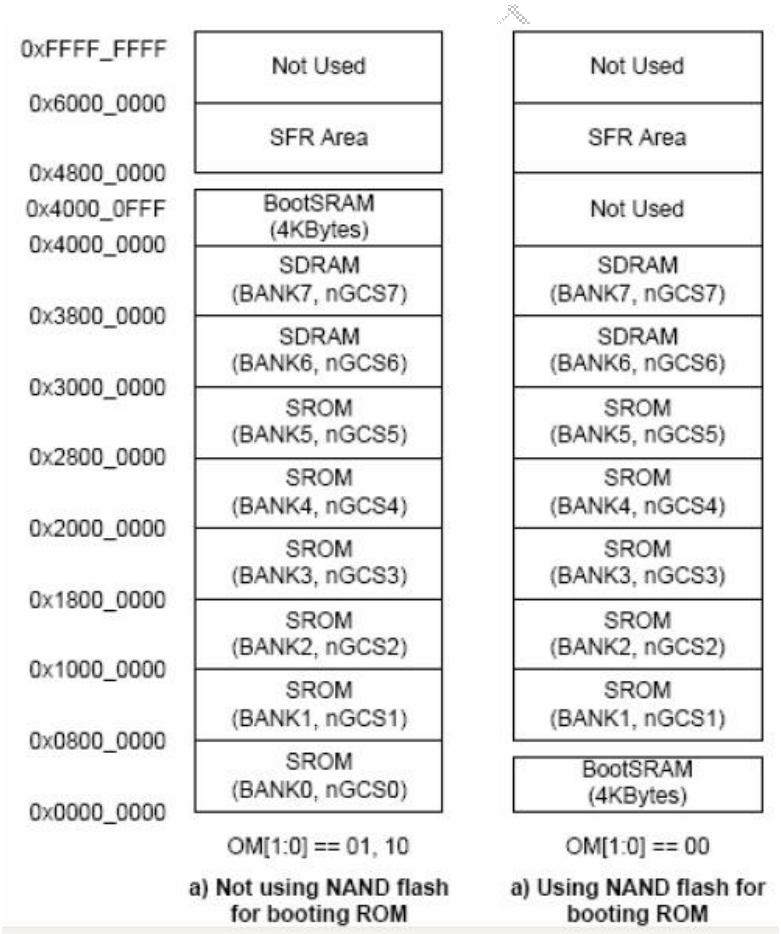

图 4.12　S3C2410 两种启动模式地址映射

3．缓冲存储器

S3C2410X 是带有指令缓存（16KB）和数据缓存（16KB）的联合缓存装置，一个缓冲区能够保持 16 字的数据和 4 个地址。

4．时钟和电源管理

S3C2410X 采用独特的时钟管理模式，它具有 PLL（相位锁定环路，用于稳定频率）的芯片时钟发生器，而在此，PLL 又分为 UPLL 和 MPLL。其中 UPLL 时钟发生器用于主/从 USB 操作，MPLL 时钟发生器用于产生主时钟，使其能在在极限频率 203MHz（1.8V）上运行。

S3C2410X 的电源管理模式又分为正常，慢速，空闲和掉电 4 种模式。其中慢速模式为不带 PLL 的低频时钟模式，空闲模式始终为 CPU 停止模式，掉电模式为所有外围设备全部掉电仅内核电源供电模式。

另外，S3C2410X 对片内的各个部件采用独立的供电方式。

- 1.8V 的内核供电。
- 3.3V 的存储器独立供电（通常对 SDRAM 采用 3.3V，对移动 SDRAM 采用 1.8/2.5V）。
- 3.3V 的 VDDQ。
- 3.3V 的 I/O 独立供电。

由于在嵌入式中电源管理非常关键，它直接涉及到功耗等各方面的系统性能，而 S3C2410X 的电源管理中独立的供电方式和多种模式可以有效地处理系统的不同状态，从而达到最优的配置。

5．中断控制

中断处理在嵌入式的开发中非常重要，尤其对于从单片机转入到嵌入式的读者来说，对比单片机中简单的中断模式而言，ARM 中的中断处理要复杂得多。如果读者无相关基础，建议先熟悉相关的基础概念再进行下一步学习。

首先给出了一般的中断处理流程，如图 4.13 所示。

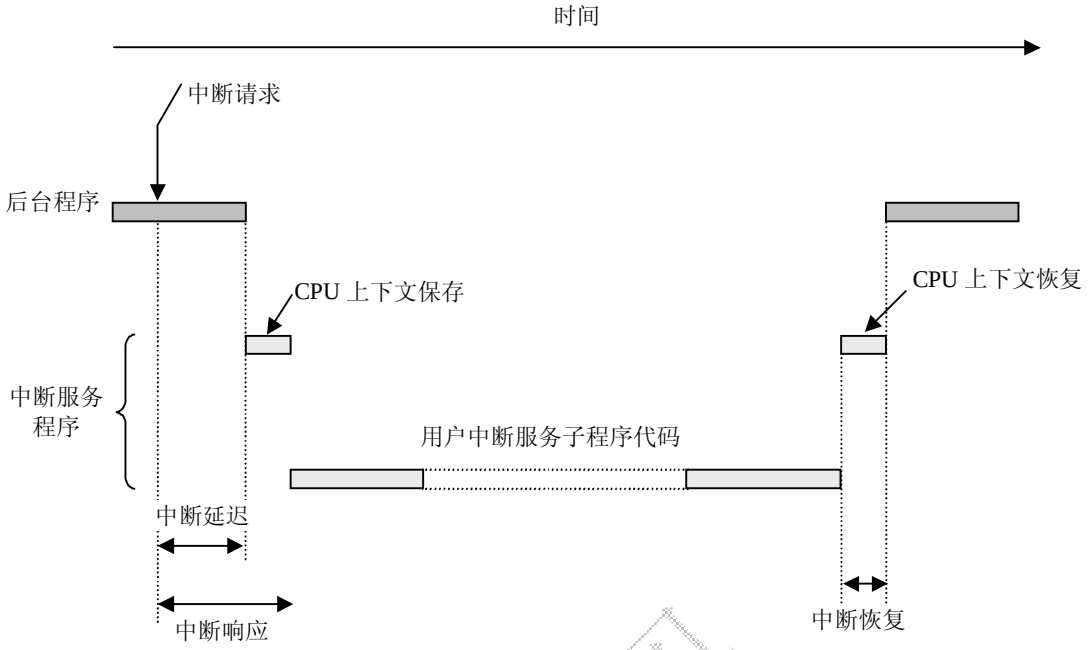

图 4.13　一般中断处理流程

S3C2410X 包括 55 个中断源，其中有 1 个看门狗定时器中断、5 个定时器中断、9 个通用异步串行口中断、24 个外部中断、4 个 DMA 中断、2 个 RTC（实时时钟控制器）中断、2 个 USB 中断、1 个 LCD 中断和 1 个电池故障。其中，对外部中断源具有电平/边沿两种触发模式。另外，对于非常紧急的中断可以支持使用快速中断请求（FIQ）。

S3C2410X 的中断处理流程（该图摘自 S3C2410X 用户手册）如图 4.14 所示。

图中的 SUBSRCPND、SRCPND、SUBMASK、MASK 和 MODE 都代表寄存器，其中 SUBSRCPND 和 SRCPND 寄存器表明有哪些中断被触发了和是否正在等待处理（pending）；SUBMASK（INTSUBMSK 寄存器）和 MASK（INTMSK 寄存器）用于屏蔽某些中断。

图中的“Request sources（with sub –register）”表示的是 INT_RXD0、INT_TXD0 等 11 个中断源，它们不同于“Request sources（without sub –register）”的操作在于：

（1）“Request sources（without sub –register）”中的中断源被触发之后，SRCPND 寄存器中相应位被置 1，如果此中断没有被 INTMSK 寄存器屏蔽、或者是快中断（FIQ）的话，它将被进一步处理。

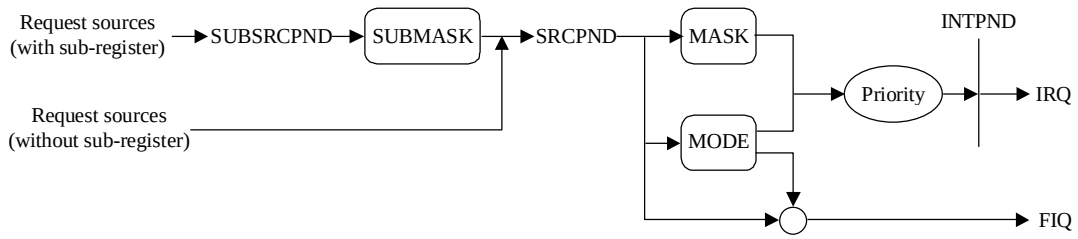

图 4.14　S3C2410X 中断处理流程

（2）对于“Request sources（with sub –register）”中的中断源被触发之后，SUBSRCPND 寄存器中的相应位被置 1，如果此中断没有被 SUBMSK 寄存器屏蔽的话，它在 SRCPND 寄存器中的相应位也被置 1。在此之后的两者的处理过程一样了。

接下来，在 SRCPND 寄存器中，被触发的中断的相应位被置 1，等待处理。

（1）如果被触发的中断中有快中断（FIQ）——MODE（INTMOD 寄存器）中为 1 的位对应的中断，则 CPU 的 FIQ 中断函数被调用。注意：FIQ 只能分配一个，即 INTMOD 中只能有一位被设为 1。

（2）对于一般中断 IRQ，可能同时有几个中断被触发，未被 INTMSK 寄存器屏蔽的中断经过比较后，选出优先级最高的中断，然后 CPU 调用 IRQ 中断处理函数。中断处理函数可以通过读取 INTPND（标识最高优先级的寄存器）寄存器来确定中断源是哪个，也可以读 INTOFFSET 寄存器来确定中断源。

4.3 嵌入式软件开发流程

4.3.1 嵌入式系统开发概述

由嵌入式系统本身的特性所影响，嵌入式系统开发与通用系统的开发有很大的区别。嵌入式系统的开发主要分为系统总体开发、嵌入式硬件开发和嵌入式软件开发 3 大部分，其总体流程图如图 4.15 所示。

在系统总体开发中，由于嵌入式系统与硬件依赖程序非常紧密，往往某些需求只能通过特定的硬件才能实现，因此需要进行处理器选型，以更好地满足产品的需求。另外，对于有些硬件和软件都可以实现的功能，就需要在成本和性能上做出抉择。往往通过硬件实现会增加产品的成品，但能大大提高产品的性能和可靠性。

再次，开发环境的选择对于嵌入式系统的开发也有很大的影响。这里的开发环境包括嵌入式操作系统的选择以及开发工具的选择等。本书在 4.1.5 节对各种不同的嵌入式操作系统进行了比较，读者可以以此为依据进行相关的选择。比如，对开发成本和进度限制较大的产品可以选择嵌入式 Linux，对实时性要求非常高的产品可以选择 Vxworks 等。

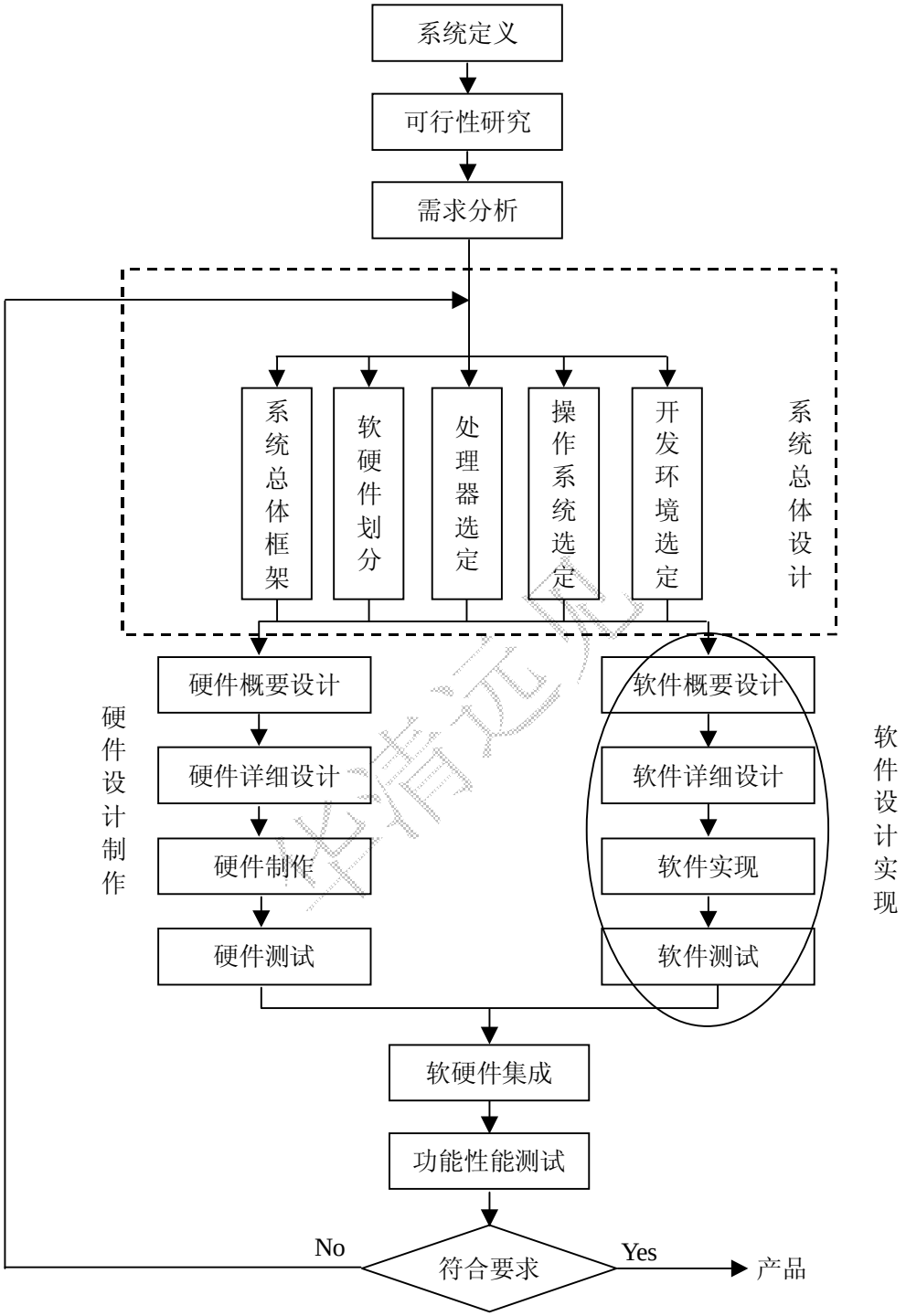

图 4.15　嵌入式系统开发流程图

由于本书主要讨论嵌入式软件的应用开发，因此对硬件开发不做详细讲解，而主要讨论嵌入式软件开发的流程。

4.3.2 嵌入式软件开发概述

嵌入式软件开发总体流程为图 4.15 中“软件设计实现”部分所示，它同通用计算机软件开发一样，分为需求分析、软件概要设计、软件详细设计、软件实现和软件测试。其中嵌入式软件需求分析与硬件的需求分析合二为一，故没有分开画出。

由于在嵌入式软件开发的工具非常多，为了更好地帮助读者选择开发工具，下面首先对嵌入式软件开发过程中所使用的工具做一简单归纳。

嵌入式软件的开发工具根据不同的开发过程而划分，如比在需求分析阶段，可以选择 IBM 的 Rational Rose 等软件，而在程序开发阶段可以采用 CodeWarrior（下面要介绍的 ADS 的一个工具）等，在调试阶段所用的 Multi-ICE 等。同时，不同的嵌入式操作系统往往会有配套的开发工具，比如 Vxworks 有集成开发环境 Tornado，WinCE 的集成开发环境 WinCE Platform 等。此外，不同的处理器可能还有针对的开发工具，比如 ARM 的常用集成开发工具 ADS 等。在这里，大多数软件都有比较高的使用费用，但也可以大大加快产品的开发进度，用户可以根据需求自行选择。图 4.16 是嵌入式开发的不同阶段的常用软件。

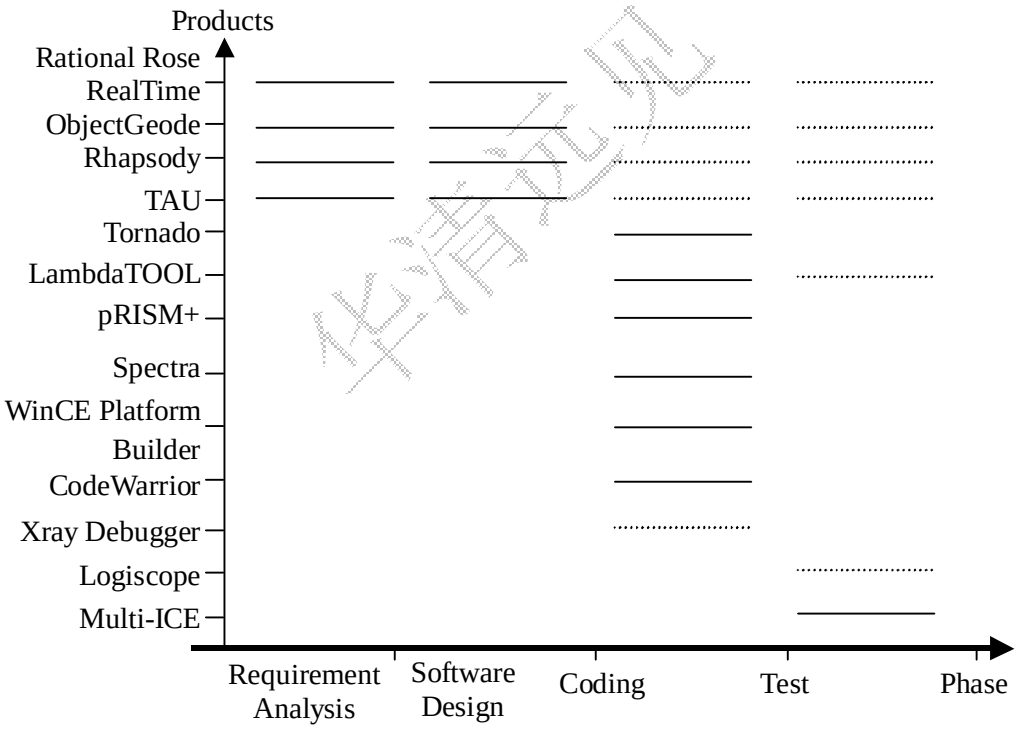

图 4.16 嵌入式开发不同阶段的常用软件

嵌入式系统的软件开发与通常软件开发的区别主要在于软件实现部分，其中又可以分为编译和调试两部分，下面分别对这两部分进行讲解。

1．交叉编译

嵌入式软件开发所采用的编译为交叉编译。所谓交叉编译就是在一个平台上生成可以在

另一个平台上执行的代码。在第 3 章中已经提到，编译的最主要的工作就在将程序转化成运行该程序的 CPU 所能识别的机器代码，由于不同的体系结构有不同的指令系统。因此，不同的 CPU 需要有相应的编译器，而交叉编译就如同翻译一样，把相同的程序代码翻译称不同的 CPU 对应语言。要注意的是，编译器本身也是程序，也要在与之对应的某一个 CPU 平台上运行。嵌入式系统交叉编译环境如图 4.17 所示。

图 4.17　交叉编译环境

小知识　与交叉编译相对应，平时常用的编译称本地编译。

这里一般把进行交叉编译的主机称为宿主机，也就是普通的通用计算机，而把程序实际的运行环境称为目标机，也就是嵌入式系统环境。由于一般通用计算机拥有非常丰富的系统资源、使用方便的集成开发环境和调试工具等，而嵌入式系统的系统资源非常紧缺，没有相关的编译工具，因此，嵌入式系统的开发需要借助宿主机（通用计算机）来编译出目标机的可执行代码。

由于编译的过程包括编译、链接等几个阶段，因此，嵌入式的交叉编译也包括交叉编译、交叉链接等过程，通常 ARM 的交叉编译器为 arm-elf-gcc，交叉链接器为 arm-elf-ld，交叉编译过程如图 4.18 所示。

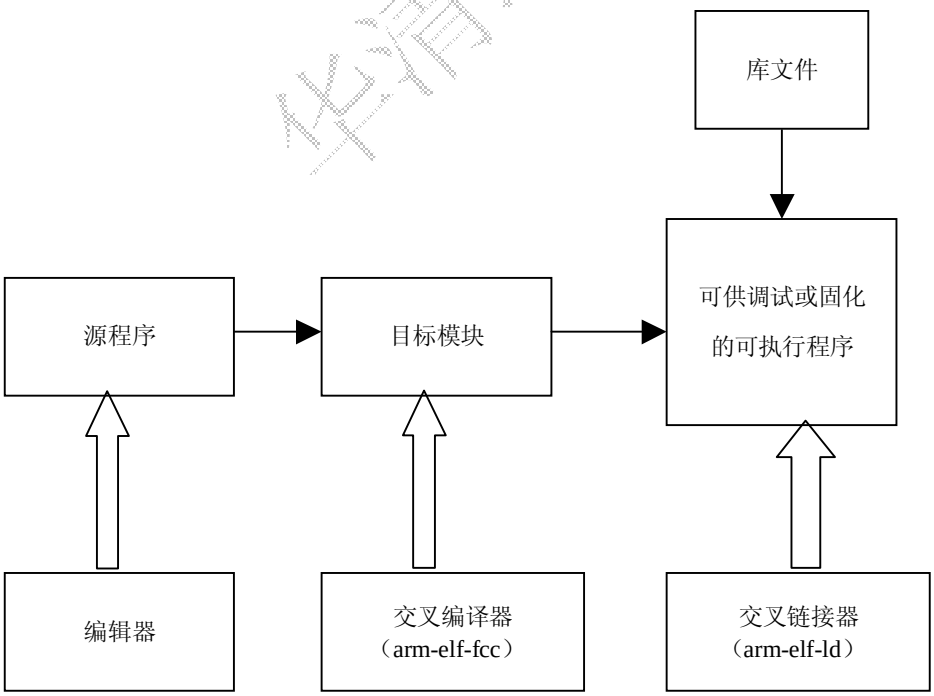

图 4.18　嵌入式交叉编译过程

2．交叉调试

嵌入式软件经过编译和链接后即进入调试阶段，调试是软件开发过程中必不可少的一个环节，嵌入式软件开发过程中的交叉调试与通用软件开发过程中的调试方式有很大的差别。在常见软件开发中，调试器与被调试的程序往往运行在同一台计算机上，调试器是一个单独运行着的进程，它通过操作系统提供的调试接口来控制被调试的进程。而在嵌入式软件开发中，调试时采用的是在宿主机和目标机之间进行的交叉调试，调试器仍然运行在宿主机的通用操作系统之上，但被调试的进程却是运行在基于特定硬件平台的嵌入式操作系统中，调试器和被调试进程通过串口或者网络进行通信，调试器可以控制、访问被调试进程，读取被调试进程的当前状态，并能够改变被调试进程的运行状态。

嵌入式系统的交叉调试有多种方法，主要可分为软件方式和硬件方式两种。它们一般都具有如下一些典型特点。

- 调试器和被调试进程运行在不同的机器上，调试器运行在 PC 机或者工作站上（宿主机），而被调试的进程则运行在各种专业调试板上（目标机）。
- 调试器通过某种通信方式（串口、并口、网络、JTAG 等）控制被调试进程。
- 在目标机上一般会具备某种形式的调试代理，它负责与调试器共同配合完成对目标机上运行着的进程的调试。这种调试代理可能是某些支持调试功能的硬件设备，也可能是某些专门的调试软件（如 gdbserver）。
- 目标机可能是某种形式的系统仿真器，通过在宿主机上运行目标机的仿真软件，整个调试过程可以在一台计算机上运行。此时物理上虽然只有一台计算机，但逻辑上仍然存在着宿主机和目标机的区别。

下面分别就软件调试桩方式和硬件片上调试两种方式进行详细介绍。

（1）软件方式

软件方式调试主要是通过插入调试桩的方式来进行的。调试桩方式进行调试是通过目标操作系统和调试器内分别加入某些功能模块，二者互通信息来进行调试。该方式的典型调试器有 Gdb 调试器。

GDB 的交叉调试器分为 GdbServer 和 GdbClient，其中的 GdbServer 就作为调试桩在安装在目标板上，GdbClient 就是驻于本地的 Gdb 调试器。它们的调试原理图如图 4.19 所示。

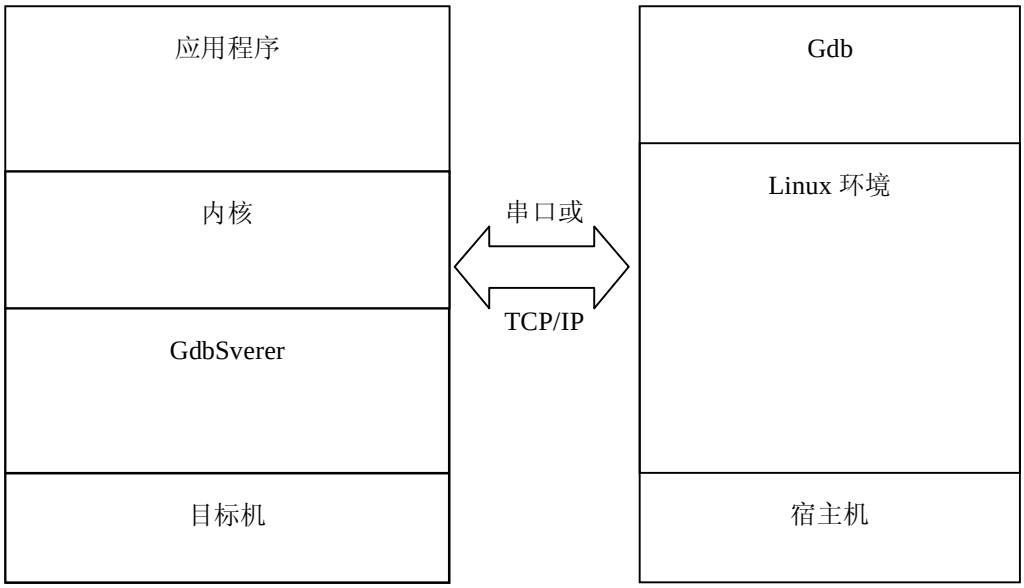

图 4.19　Gdb 远程调试原理图

Gdb 调试桩的工作流程。

- 首先，建立调试器（本地 Gdb）与目标操作系统的通信连接，可通过串口、网卡、并口等多种方式。
- 然后，在目标机上开启 Gdbserver 进程，并监听对应端口。
- 在宿主机上运行调试器 Gdb，这时，Gdb 就会自动寻找远端的通信进程，也就是 Gdbserver 的所在进程。
- 在宿主机上的 Gdb 通过 Gdbserver 请求对目标机上的程序发出控制命令。这时，Gdbserver 将请求转化为程序的地址空间或目标平台的某些寄存器的访问，这对于没有虚拟存储器的简单的嵌入式操作系统而言，是十分容易的。
- Gdbserver 把目标操作系统的所有异常处理转向通信模块，并告知宿主机上 Gdb 当前异常。
- 宿主机上的 Gdb 向用户显示被调试程序产生了哪一类异常。

这样就完成了调试的整个过程。这个方案的实质是用软件接管目标机的全部异常处理及部分中断处理，并在其中插入调试端口通信模块，与主机的调试器进行交互。但是它只能在目标机系统初始化完毕、调试通信端口初始化完成后才能起作用，因此，一般只能用于调试运行于目标操作系统之上的应用程序，而不宜用来调试目标操作系统的内核代码及启动代码。而且，它必须改变目标操作系统，因此，也就多了一个不用于正是发布的调试版。

（2）硬件调试

相对于软件调试而言，使用硬件调试器可以获得更强大的调试功能和更优秀的调试性能。硬件调试器的基本原理是通过仿真硬件的执行过程，让开发者在调试时可以随时了解到系统的当前执行情况。目前嵌入式系统开发中最常用到的硬件调试器是 ROMMonitor、ROMEmulator、In-CircuitEmulator 和 In-CircuitDebugger。

- 采用 ROMMonitor 方式进行交叉调试需要在宿主机上运行调试器，在目标机上运行

ROM 监视器（ROMMonitor）和被调试程序，宿主机通过调试器与目标机上的 ROM 监视器遵循远程调试协议建立通信连接。ROM 监视器可以是一段运行在目标机 ROM 上的可执行程序，也可以是一个专门的硬件调试设备，它负责监控目标机上被调试程序的运行情况，能够与宿主机端的调试器一同完成对应用程序的调试。

在使用这种调试方式时，被调试程序首先通过 ROM 监视器下载到目标机，然后在 ROM 监视器的监控下完成调试。

优点：ROM 监视器功能强大，能够完成设置断点、单步执行、查看寄存器、修改内存空间等各项调试功能。

确定：同软件调试一样，使用 ROM 监视器目标机和宿主机必须建立通信连接。

其原理图如图 4.20 所示。

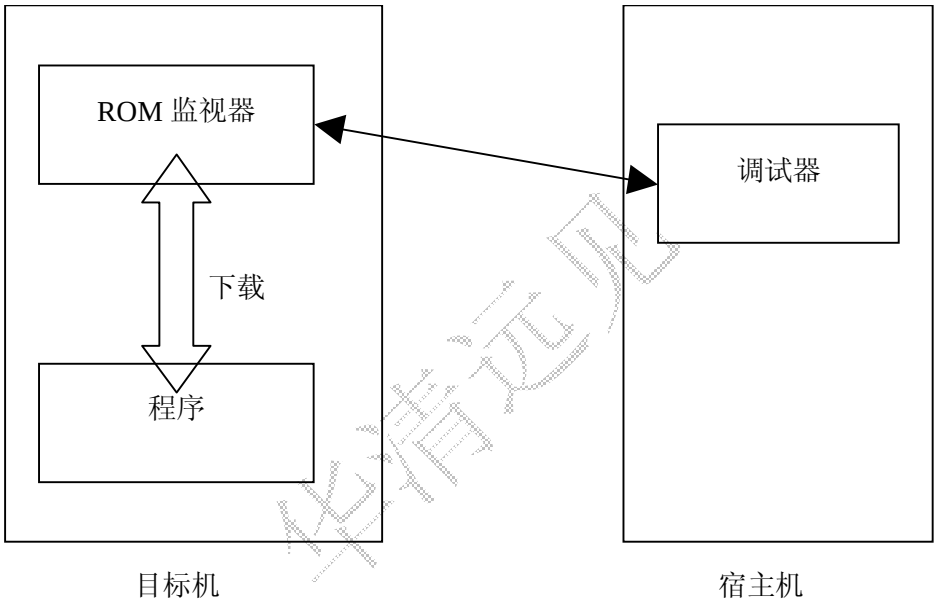

图 4.20 ROMMonitor 调试方式

- 采用 ROMEmulator 方式进行交叉调试时需要使用 ROM 仿真器，并且它通常被插入到目标机上的 ROM 插槽中，专门用于仿真目标机上的 ROM 芯片。

在使用这种调试方式时，被调试程序首先下载到 ROM 仿真器中，因此等效于下载到目标机的 ROM 芯片上，然后在 ROM 仿真器中完成对目标程序的调试。

优点：避免了每次修改程序后都必须重新烧写到目标机的 ROM 中。

缺点：ROM 仿真器本身比较昂贵，功能相对来讲又比较单一，只适应于某些特定场合。

其原理图如图 4.21 所示。

- 采用 In-CircuitEmulator（ICE）方式进行交叉调试时需要使用在线仿真器，它是目前最为有效的嵌入式系统的调试手段。它是仿照目标机上的 CPU 而专门设计的硬件，可以完全仿真处理器芯片的行为。仿真器与目标板可以通过仿真头连接，与宿主机可以通过串口、并口、网线或 USB 口等连接方式。由于仿真器自成体系，所以调试时既可以连接目标板，也可以不连接目标板。

在线仿真器提供了非常丰富的调试功能。在使用在线仿真器进行调试的过程中，可以按

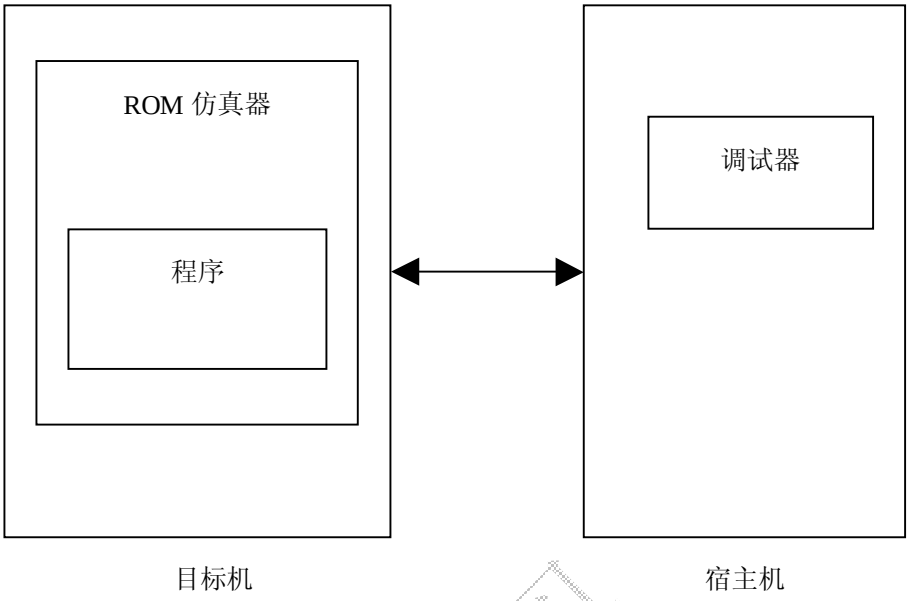

图 4.21　ROMEmulator 调试方式

顺序单步执行，也可以倒退执行，还可以实时查看所有需要的数据，从而给调试过程带来了很多的便利。嵌入式系统应用的一个显著特点是与现实世界中的硬件直接相关，并存在各种异变和事先未知的变化，从而给微处理器的指令执行带来各种不确定因素，这种不确定性在目前情况下只有通过在线仿真器才有可能发现。

优点：功能强大，软硬件都可做到完全实时在线调试。

确定：价格昂贵。

其原理图如图 4.22 所示。

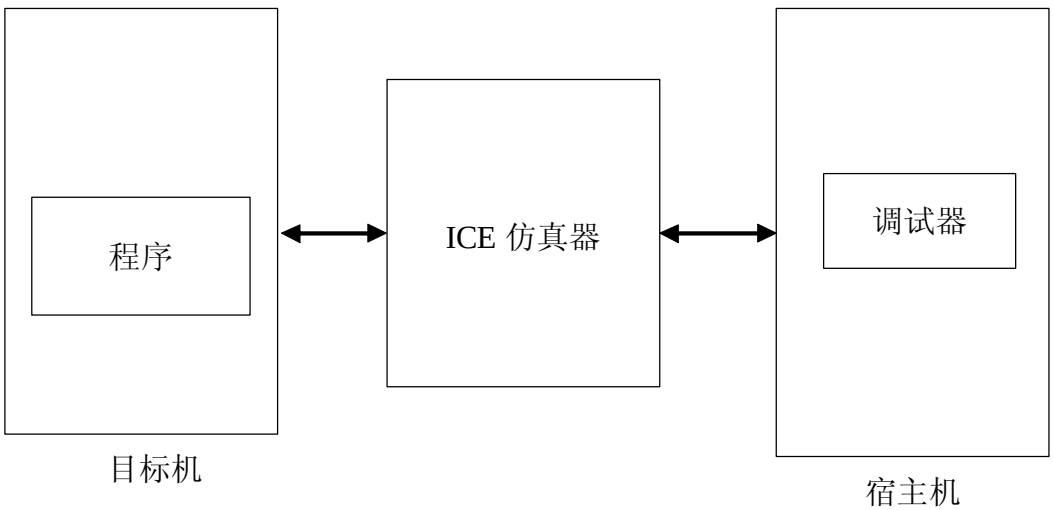

图 4.22　ICE 调试方式

- 采用 In-CircuitDebugger（ICD）方式进行交叉调试时需要使用在线调试器。由于 ICE 的价格非常昂贵，并且每种 CPU 都需要一种与之对应的 ICE，使得开发成本非常高。一个比较好的解决办法是让 CPU 直接在其内部实现调试功能，并通过在开发板上引出的调试端口发送调试命令和接收调试信息，完成调试过程。如在采用非常广泛的 ARM 处理器的 JTAG 端口技术就是由此而诞生的。

JTAG 是 1985 年指定的检测 PCB 和 IC 芯片的一个标准。1990 年被修改成为 IEEE 的一个标准，即 IEEE1149.1。JTAG 标准所采用的主要技术为边界扫描技术，它的基本思想就是在靠近芯片的输入输出管脚上增加一个移位寄存器单元。因为这些移位寄存器单元都分布在芯片的边界上(周围)，所以被称为边界扫描寄存器(Boundary-Scan Register Cell)。

当芯片处于调试状态时候，这些边界扫描寄存器可以将芯片和外围的输入输出隔离开来。通过这些边界扫描寄存器单元，可以实现对芯片输入输出信号的观察和控制。对于芯片的输入脚，可通过与之相连的边界扫描寄存器单元把信号（数据）加载到该管脚中去；对于芯片的输出管脚，可以通过与之相连的边界扫描寄存器单元“捕获”（CAPTURE）该管脚的输出信号。这样，边界扫描寄存器提供了一个便捷的方式用于观测和控制所需要调试的芯片。

现在较为高档的微处理器都带有 JTAG 接口，包括 ARM7、ARM9、StrongARM、DSP 等，通过 JTAG 接口可以方便地对目标系统进行测试，同时，还可以实现的 Flash 的编程，是非常受人欢迎的。

优点：连接简单，成本低。

缺点：特性受制于芯片厂商。

其原理图如图 4.23 所示。

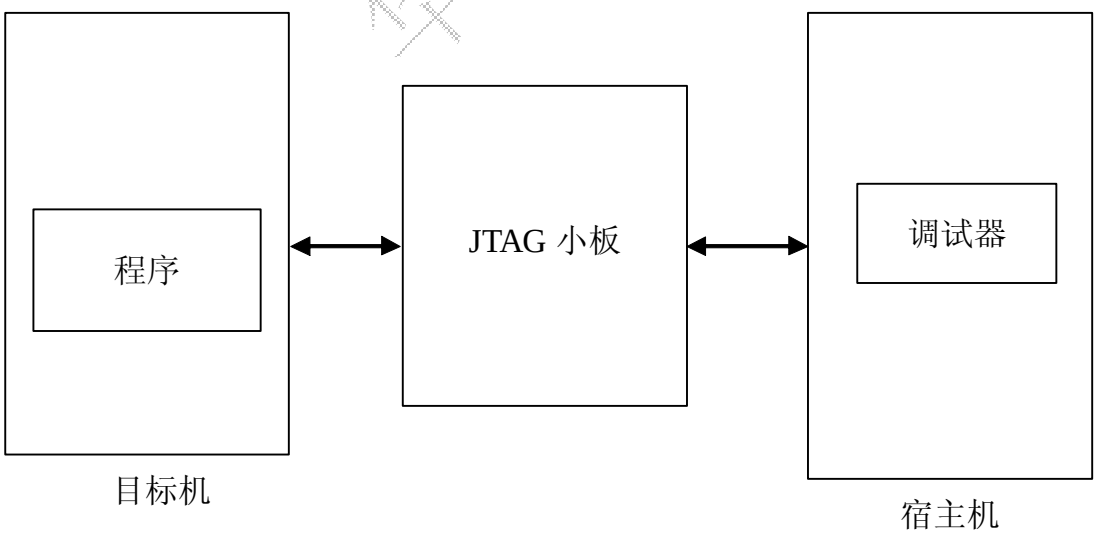

图 4.23　JTAG 调试方式

4.4 实验内容——使用 JTAG 烧写 NAND Flash

1. 实验目的

通过使用 JTAG 烧写 Flash 的实验，了解嵌入式硬件环境，熟悉 JTAG 的使用，为今后的进一步学习打下良好的基础。（本书以优龙的 FS2410 及 Flash 烧写工具为例进行讲解，不同厂商的开发板都会提供相应的 Flash 烧写工具，并有相应的说明文档，请读者在了解基本原理之后查阅相关手册）

2. 实验内容

（1）熟悉开发板的硬件布局。
（2）连接 JTAG 口。
（2）安装 giveio（用于烧写 Flash）驱动。
（3）打开 SJF2410_BIOS.BAT（Flash 烧写程序）进行烧写。

3. 实验步骤

（1）熟悉开发板硬件设备请参阅本章 4.2 节的 FS2410 实物图。
（2）用 20 针的排线将 20 针的 JTAG 接口与 JTAG 小板的 JP3 接口相连。
（3）安装 givieo 驱动，如图 4.24 所示。

图 4.24　安装 givieo

单击 Install 按钮，出现如图 4.25 所示，就表明驱动安装成功。

图 4.25 givieo 驱动安装完成

（4）打开 SJF2410_BIOS.BAT，如图 4.26 所示。

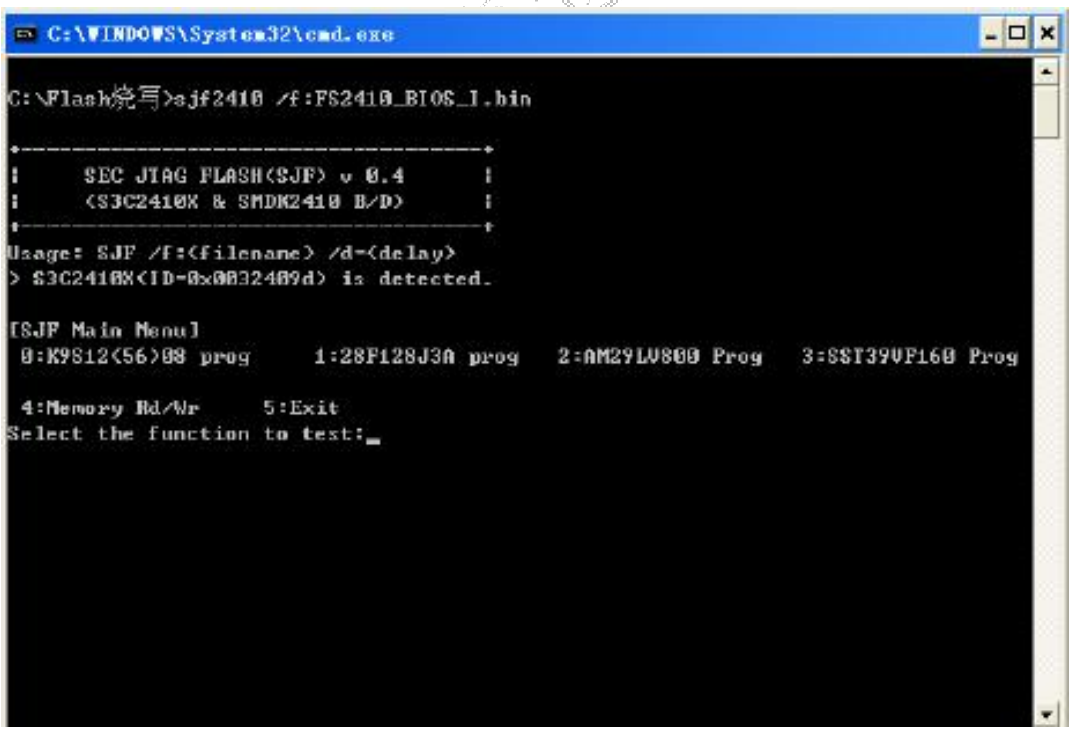

图 4.26 打开 SJF2410_BIOS.BAT

（5）在“Select the function to test:”输入“0”，表示对 K9S1208（FS2410 的 NAND Flash 的芯片型号）进行烧写，如图 4.27 所示。

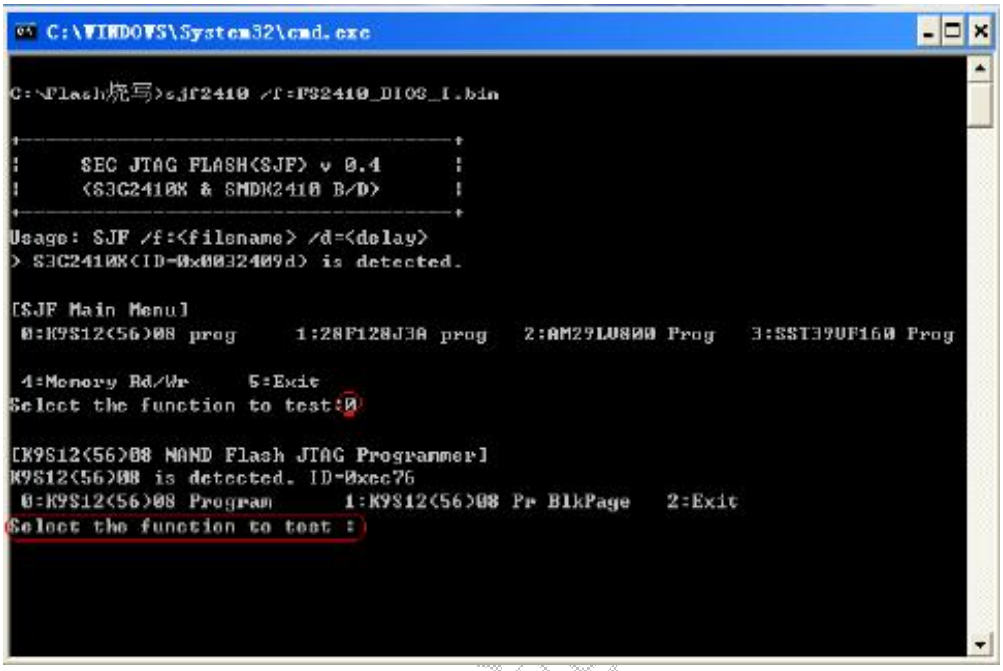

图 4.27　选择烧写对应芯片

（6）在接下来的“Select the function to test:”里输入“0”，表示烧写类型为程序。再在接下来的“Input the target block”里输入希望的偏移地址，在此处写为“0”，如图 4.28 所示。

（7）接下来，Flash 完成烧写过程，如图 4.29 所示。

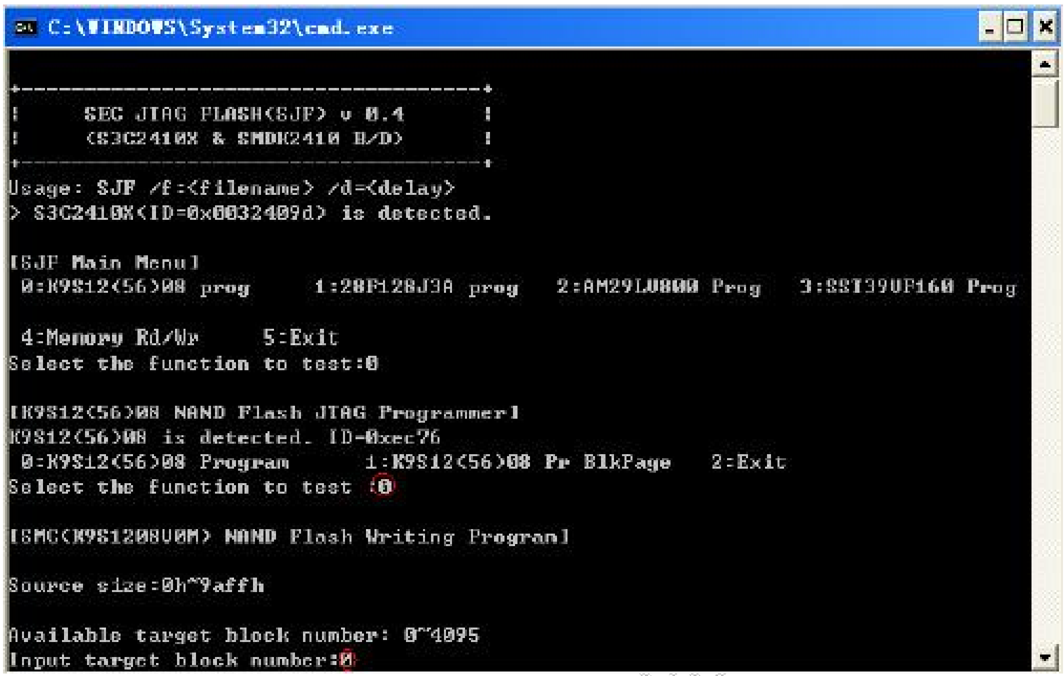

图 4.28 选择烧写类型及偏移地址

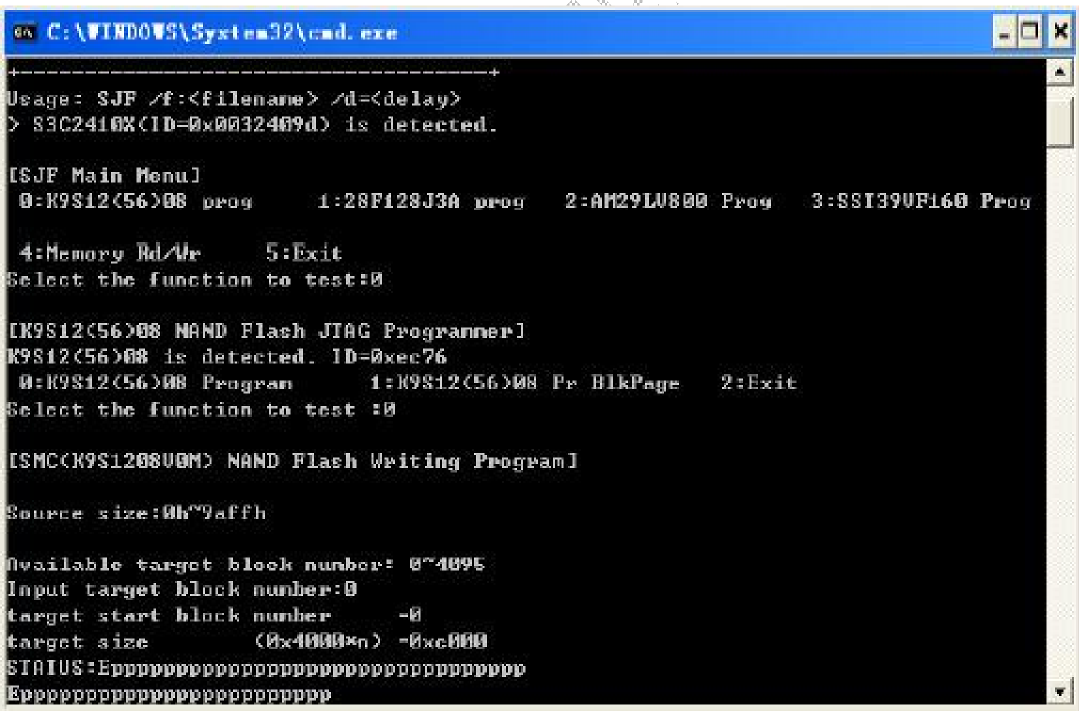

图 4.29 Flash 烧写过程

4．实验结果

系统烧写完成后，程序会自动退出，这样就完成了对 Flash 的烧写。

本 章 小 结

本章讲解了嵌入式中的基本概念，包括嵌入式系统的含义、它的发展历史、嵌入式系统的特点以及嵌入式系统的体系结构。在这里，重点要掌握嵌入式系统和通用计算机的区别，希望读者能对以上几方面都能对其进行一一比较，以加深对嵌入式系统的理解。

接下来对 ARM 体系进行了概括性地讲解，希望读者能重点掌握 ARM9 的特性，有条件的读者希望能结合实际开发板进行学习，没有开发板的读者也可参看图中的实物图，以获得感性的认识。另外，不同的硬件平台都会有一定的区别，但其主要原理是一样的，对于某些细节的不同处理请读者参阅不同厂商的用户手册。

本章的最后讲解了嵌入式软件开发的流程，其中重点讲解了交叉编译和交叉调试，这些概念初次学习会感觉比较枯燥，但这些概念又是非常重要的，在后面的具体开发中会经常涉及到，希望读者对这些内容能够认真消化。

最后安排的一个实验希望有条件的读者能动手做做，当然在做之前一定认真阅读不同厂商提供的用户手册。

思考与练习

1．从各方面比较嵌入式系统与通用计算器的区别。
2．ARM9 有哪些优于 ARM7 的特性？
3．什么是交叉编译？为什么要进行交叉编译？
4．嵌入式开发的常用调试手段有那几种？说出它们各自的优缺点。