# ARM 启动原理分析与实现

洛阳师范学院计算机系　郝红旗　田　斌

[摘　要]本文介绍了启动代码的作用,以及启动代码中的向量表定义、堆栈初始化、系统硬件初始化,然后重点分析了分散加载机制的目的和实现。并利用模块化设计给出了一个基于LPC2210处理器启动代码的实现。

## 引言

随着ARM芯片逐渐占据32位市场,越来越多的开发人员开始转向使用ARM芯片。但是ARM公司只设计内核,并不生产芯片,只是把内核授权给其他厂商。其他厂商购买了授权后加入自己的外设,生产出各具特色的芯片。这样一方面促进了基于ARM处理器核芯片的多元化,另一方面使各种芯片的启动代码差别较大,不易编写统一的启动代码。但是启动代码还是有些共性的,因此了解这些共性,对开发具体的应用至关重要。

## 启动原理

在32位的ARM芯片的程序开发中,一般要采用C语言编程,在运行C程序之前,要对系统初始化,就像PC机的BIOS一样,这部分代码就是启动代码。启动代码一般要实现以下功能:异常向量表的定义、各模式堆栈的初始化、系统变量初始化、中断系统初始化和地址重映射等。

ARM芯片复位后,系统进入管理模式,ARM状态,PC(R15)寄存器的值为0x00000000。因此必须保证用户的向量表代码定位在0x00000000处。但是可以通过分散加载以及重映射技术把这部分代码映射到具体的位置。

本试验板采用LPC2210芯片,由于其片上只有16KB的RAM,资源有限,所以外扩了两片存储器,一片作为FLASH,另一片作为RAM。这样存储器空间分配为:

0×40000000～0×40003fff 片内RAM

0×80000000～0×8007ffff 外扩RAM

0×81000000～0×8107ffff 外扩FLASH

本文只以调试状态的分散加载和重映射来说明,所以只用到了片内RAM和外扩RAM,参考本文,可以很容易写出FLASH的代码。分散加载描述文件如下:

```
ROM_LOAD 0x80000000
{
ROM_EXEC 0x80000000
{
vectors.o(Vect,+First)
*(+RO)
}
ERAM 0x80040000
{
*(+RW,+ZI)
}
HEAP +0 UNINIT
{
heap.o(+ZI)
}
STACKS 0x40004000 UNINIT
{
stacks.o(+ZI)
}
}
```

## 建立中断向量表

系统上电复位后,首先执行0x80000000处的代码,结果是将外部RAM 0x80000000重映射(REMAP)为逻辑地址0x00000000,或者说当系统执行0x0000000处的代码时,实际执行的是0x80000000处的代码。因此对0x80000000处程序的编写,是我们编程的开始,这部分开始的32个字节就是中断向量表。这样当异常发生时,系统自动跳转到相应的异常执行程序。例如,当有外部中断发生时,系统就跳到0x18执行程序。

中断向量表中,0x00～0x1C的这段程序只存放每个中断的入口,程序从这个入口跳转到各自的异常处理程序。建立中断向量表的代码如下:

```
Reset
LDR PC,ResetAddr;跳转到程序复位
LDR PC,UndefinedAddr;跳转到未定义指令
LDR PC,SWI_Addr;跳转到软件中断
LDR PC,PrefetchAddr;跳转到指令预取异常
LDR PC,DataAbortAddr;跳转到数据异常
DCD 0xb9205f80;保留
LDR PC,[PC,#-0xff0];跳转到IRQ中断
LDR PC,FIQ_Addr;跳转到FIQ中断
ResetAddr DCD ResetInit
UndefinedAddr DCD Undefined
SWI_Addr ……
```

程序正常复位后,首先执行"LDR PC,ResetAddr",这条指令的含义是把ResetAddr的值加载到程序计数器(PC),而ResetAddr的值通过后面的一条伪指令"ResetAddr DCD ResetInit"得到了定义。

## 初始化各模式堆栈

ARM处理器共有7中模式:

①用户模式 正常程序工作模式,不能切换到其他模式

②快速中断模式 支持高速数据处理及通道处理,FIQ异常响应时,进入此模式

③中断模式 用于通用中断处理,IRQ响应时,进入此模式

④管理模式 操作系统保护代码,系统复位和软件中断响应时,进入此模式

⑤中止模式 用于虚拟内存/存储保护

⑥未定义模式 支持硬件协处理器的软件仿真,未定义异常响应时,进入此模式

⑦系统模式 用于支持操作系统的特权任务等,与用户模式类似,但具有特权

ARM在不同的工作模式下,系统所要求的堆栈空间不同,因此要分别进行初始化。程序代码如下:

```
InitStack
LDR R0,=bottom_of_heap
;设置管理模式堆栈
MSR CPSR_c,#0xd3
SUB SP,R1,#Offset_SVC_Stack
;设置中断模式堆栈
……
;设置系统模式堆栈
MSR CPSR_c,#0xdf
LDR SP,=StackUsr
```

本设计把各个模式的堆栈放在了堆的下面(=bottom_of_heap),有了这个参考地址,就可以根据各个模式堆栈的偏移进行堆栈设置。各模式堆栈的偏移是这样计算的:前一个模式堆栈偏移+前一个模式堆栈的长度。代码如下:

```
Offset_SVC_Stack EQU 0
Offset_FIQ_Stack EQU Offset_SVC_Stack+SVC_STACK_LEGTH
Offset_IRQ_Stack EQU Offset_FIQ_Stack+FIQ_STACK_LEGTH
Offset_ABT_Stack EQU Offset_IRQ_Stack+IRQ_STACK_LEGTH
```

Offset __UND __Stack EQU Offset __ABT __Stack+ ABT __STACK __LEGTH

这样通过下面分散加载描述文件:

```
ERAM 0x80040000
{
* (+ RW, + ZI)
}
```

就可以把各模式堆栈初始化到外部存储器。

**系统硬件初始化**

这部分代码主要用来初始化锁相环, 外部存储控制器(EMC), 向量中断控制器等。至此, 初始化就基本完成, 通过"B __main "命令就可以跳转到用户的C 语言主程序。

**初始化C 语言库**

为了使用C 语言进行开发, 就必须初始化C 库使用的堆和栈。这里需要注意的是一个嵌入式系统不能使用ADS 的semihost 功能, 所以要通过 IMPORT —— use __ no __ semihosting __swi 来禁用。另外还要注意函数 —— user __initial __stackheap, 在重新定义后, 必须导出, 否则系统就会使用C 库中默认的。该部分代码如下:

```
IMPORT —— use __no __ semihosting __ swi
EXPORT —— user __ initial __ stackheap
—— user __ initial __ stackheap
LDR r0, = bottom __of __heap
MOV pc, lr
```

**程序的整体设计和装载**

根据上述对ARM 启动代码的分析, 依据模块化思想, 将本设计划分为如下几个模块:

Vectors. s 完成向量表的设置

Init. s 完成各模式堆栈初始化, 系统硬件初始化, 呼叫__main 函数

Heap. s 设置C 语言使用的堆

Stacks 设置C 语言使用的栈

根据上述设计, 通过分散加载机制, 可得到运行时存储器映射如图1 所示:

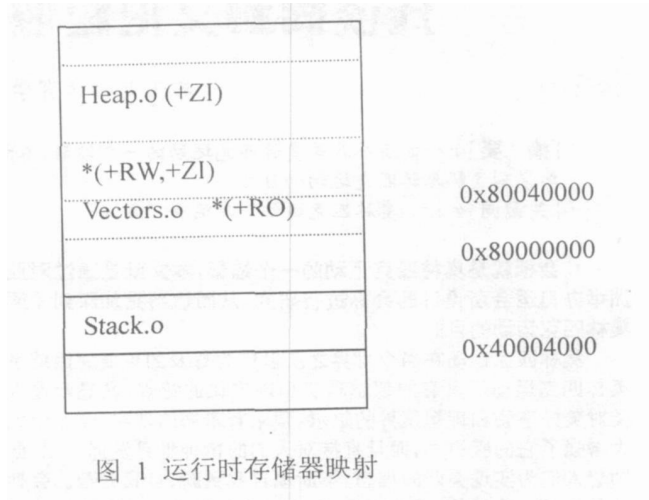


图 1 运行时存储器映射

可以根据不同的硬件设计, 参照本实现修改分散加载描述文件, 达到不同的应用。

**结论**

通过对ARM 启动的分析与设计, 不仅可以深入理解ARM 处理器编程, 而且可以为编写复杂的应用系统提供一个牢固的底层基础。

**参考文献**

[1]周立功. ARM 嵌入式系统基础教程. 北京航空大学出版社, 2005. 1

[2]杜春雷. ARM 体系结构与编程. 清华大学出版社, 2003. 2

[3]ARM 公司. ADS developer guide, ADS 参考

(上接第173 页)一旦改变(例如用户的IP 地址)就要在路由器上相应地增加或修改配置, 显然, 无法达到接入型VPN 地需求。可一通过对PPTP、L2TP 和IPSec 等作修改达到该目的。

(4) IPv6 与IPv4 网络之间的NAT- PT 实验。其目的是使IPv6 节点与IPv4 节点通信时无须使用双协议栈, 这将是IPv6 向IPv4 平滑过渡的一个重要策略, 其难度在于它不仅仅是转换地址或将IPv6 与IPv4 的包头调换, 更关键的是它涉及到一系列协议(例如ICMPv6、PPP、FTP 等)也要作相应修改。

(5)Mobile IP &Mobile IPv6 的研发实验。Mobile IP 本身就是一个前沿的网络技术, 是IP 与移动通信相结合的一项重要攻关项目, 目的要实现用户在移动的过程中无须修改其IP 地址, 用户从一个网段移动到另一个网段之间的过渡处理对用户来说是透明和平滑的。Mobile IP 还有很多需要研究的课题, 例如,Mobile 环境下的网络安全, 如何在Mobile 环境下实现组播等。

(6) IPv6 下的QoS 控制实验。IPv6 增加了流标签, IPv6 小组坚信通过这一改善, 可以使得IPv6 承载业务以及QoS 控制的能力可以媲美ATM。目前, 如何对流标签和优先级的有效利用, IPv6 小组只是给出了建议性指引, 还远未达到制定标准的地步, 通过构建IPv6 overMPLS 实验系统对IPv6 环境下如何有效保证在网络业务的QoS 进行初步的研究。

希望通过实验网系统, 以技术研讨及讲座的形式介绍网络热点技术。引导学生以探索的方式学习, 指导学生在围绕IPv6 作一些研究并撰写论文, 提高他们的理论知识水平和实践能力, 在此过程中也希望在实验教学方面为提高学生的创新能力积累一些经验, 为教学改革提供一些数据依据。

**参考文献**

[1]荆山, 孙润元, 陈贞翔, 杨波. IPv4/IPv6 过渡方案的研究与应用网络部署. 通信学报, 2006, 11: 34

[2]徐继恒. 校园网IPv6 部署方案. 网络导航, 2004, 2: 9, 10

[3]周运, 赵正文. 基于IPv6 的下一代网络技术研究. 中国教育网, 2004, 1: 22

[4]何涛等. IPv6 试验床的实践与研究. 计算机应用, 2001, 12