

一种基于 TI5416 的应用程序 boot 方法

张梅娟, 石乔林, 俞光耀, 李晓磊

(中国电子科技集团公司第 58 研究所, 江苏 无锡 214035)

摘 要: 利用 TI5416 处理器可自定义 boot 程序的功能, 灵活的实现了对复杂语音程序的上电自举加载. 利用 TI5416 存储结构特点, 设计了地址译码, 实现了程序页共享, 有效节省了存储空间容量需求.

关键词: 页共享; boot 方法; TI5416

中图分类号: TP23

文献标识码: A

文章编号: 1000-7180(2016)02-0068-04

The Boot Method for the Application Program of TI5416

ZHANG Mei-juan, SHI Qiao-lin, YU Guang-yao, LI Xiao-lei

(No. 58th Research Institute of China Electronics Technology Group Corporation, Wuxi 214035, China)

Abstract: With the user-defined boot function of the TI5416, this paper provides a flexible self-triggered method for the complicated phonetic code. This paper also designs a sufficient method, according to the characteristic of TI5416 storage structure, to achieve the requirement of saving storage space with the use of page-share mode. Most of all, the technique of storage space also can apply into other processors, which is significance in related fields.

Key words: page share; boot method; TI5416

1 引言

Boot loader 是嵌入式系统在加电后执行的第一段代码, 它完成 CPU 和相关硬件的初始化之后, 再将固化的嵌入式应用程序装载到内存中, 然后跳转到应用程序所在的内存空间的入口地址, 启动应用程序运行.

对于嵌入式系统, Boot loader 是基于特定硬件平台来实现的. 因此, 几乎不可能为所有的嵌入式系统建立一个通用的 boot loader, 不同的处理器架构都有不同的 boot loader^[1]. Boot loader 不但依赖于 CPU 的体系结构, 而且依赖于嵌入式系统板级设备的配置.

Boot loader 是嵌入式系统开发的关键部分. Boot loader 的操作模式一般分为两种, 一是自启动模式, 在这种模式下, boot loader 从目标机上的某个固态存储设备上将应用程序加载到 RAM 中运行, 整个过程并没有用户介入. 二是交互模式, 在这种模式

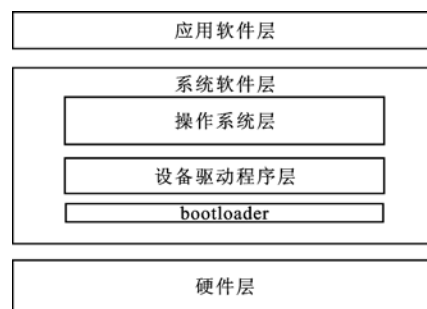


图 1 嵌入式系统层次结构

下, 目标机上的 boot loader 将通过串口或网络等通信手段从开发主机 (Host) 上下载应用程序到 RAM 中. 可以被 boot loader 写到目标机上的固态存储器中, 或者直接进入系统的引导. 也可以通过串口接收用户的命令^[2].

本文以自启动语音应用程序为目的, 提出了一种基于 TI5416 的应用程序 boot 方法, 在无用户介入的情况下, 成功的将语音应用程序装载运行起来.

收稿日期: 2015-04-16; 修回日期: 2015-05-27

基金项目: 中国电子科技集团预研项目 (51308010205)

2 TI5416 存储结构

设计 boot 程序,需要充分考虑处理器的存储结构. TMS320VC5416(简称 TI5416)是 TI 公司的一款 16 位定点 DSP 芯片,具有低功耗、实时信号处理速度快、操作灵活的特点. 已被广泛应用于无线电通信系统及语音应用系统中.

TI5416 包含 3 个独立的存储空间,程序、数据和 I/O 空间. 程序空间的可寻址范围为 $8\text{ M} \times 16\text{ bit}$;数据空间的可寻址范围为 $64\text{ k} \times 16\text{ bit}$,I/O 空间的可寻址范围为 $64\text{ k} \times 16\text{ bit}$. TI5416 片上存储器包含 $16\text{ k} \times 16\text{ bit}$ 可屏蔽 ROM, $64\text{ k} \times 16\text{ bit}$ SARAM, $64\text{ k} \times 16\text{ bit}$ DARAM. ROM 存放了已固化的 boot loader 程序,ROM 是否映射在内存空间,由 PMST 寄存器的 MP/MC 位决定;SARAM 用于存放应用程序和数据,DARAM 读写速度快,常用作栈空间和存放常用数据. SARAM 和 DARAM 均以 $8\text{ k} \times 16\text{ bit}$ 为单位分为 8 块,分别为 SARAM0 ~ SARAM7, DARAM0 ~ DARAM7. TI5416 的程序空间采用分页机制,可寻址的 $8\text{ M} \times 16\text{ bit}$ 空间以 $64\text{ k} \times 16\text{ bit}$ 大小为一页,共分 128 页. 页寻址寄存器 XPC ($\text{XPC} = 0 \sim 127$) 确定选择的页,程序计数器 PC 仅在每页的 64 k 地址范围内跳转^[3].

3 boot 程序设计

Boot 方法设计以自启动语音应用程序为目的. 本文所述的应用程序是一种低速率语音压缩算法程序,该算法程序具有运算量大,代码数据量大的特点,程序代码和数据大小约 $256\text{ k} \times 16\text{ bit}$. 代码和数据应尽量放在速度比较快的片上存储器上执行,否则很难达到实时处理的效果. TI5416 处理器片上无大的固态存储空间,因此要考虑将代码和数据放在片外固态存储器上,采用 boot 的方法,自举加载语音程序. TI5416 处理器片上的 ROM 空间内芯片生产商已固化了 boot loader,支持并行加载、I/O 加载、串口加载和 HPI 加载等多种自举加载方式. 但是这些自举加载方法牵涉到了外设接口和外围芯片,用户需要严格遵守相关的设计要求,才能成功自举加载相应程序,使用不灵活^[4]. 基于这个问题,本文提出了一种灵活的上电自举加载应用程序的方法.

3.1 设计思路

TI5416 的处理器模式状态寄存器 PMST 的 MP/MC 位值 0/1 决定了芯片是工作在微计算机模式/微处理器模式,决定了是否屏蔽片上 ROM 存

储器,从而决定使用片上 boot loader,还是需要用户自己编写 boot 程序^[5]. 本文是用户自己编写的 boot 程序,因此上电/芯片复位时,MP/MC 应为 1,芯片工作在微处理器模式. 语音程序的代码和数据均放在 $256\text{ k} \times 16\text{ bit}$ 大小的片外固态存储器上. 实现步骤:

- (1)分析空间需求,分析处理器存储空间结构,合理分配空间;
- (2)合理设计启动第一条指令的存放位置;
- (3)将代码和数据分别放入设计好的片外空间上;
- (4)设计合理有效的代码和数据搬移手段.

3.2 实现过程

语音程序需要 $256\text{ k} \times 16\text{ bit}$ 的存储空间,为避免反复更改地址映射引起不必要的错误,将 $256\text{ k} \times 16\text{ bit}$ 的片外存储空间映射在程序空间的 Page4/5/6/7/8/a/b 的高 32 k 地址空间,这些地址空间映射关系不受任何寄存器位的影响(不受 PMST 的 OV-LY/ROM 等位的影响). 程序代码和数据分块放在这些地址空间中.

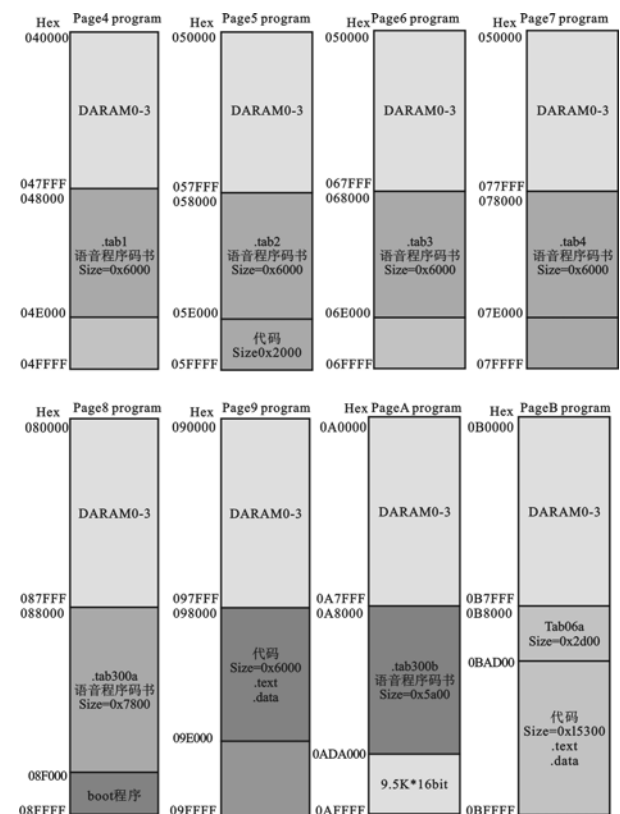


图 2 外部存储空间分配

表 1 程序各段内容说明

Mem	StartAddr	EndAddr	Size	说明
-----	-----------	---------	------	----

Page4	0x48000	0x4e000	0x6000	程序数据
Page5	0x58000	0x5e000	0x6000	程序数据
	0x5e000	0x5fff	0x2000	代码
Page6	0x68000	0x6e000	0x6000	数据
Page7	0x78000	0x7e000	0x6000	数据
	0x7e000	0x7fe00	0x1e00	数据
	0x88000	0x8f800	0x7800	代码数据
Page8	0x8f800	0x8f800	0x2cc	Boot 代码
	0x8ff80	0x8ff84	0x4	上电跳转
Page9	0x98000	0x9d800	0x6000	代码数据
PageA	0xa8000	0xada00	0x5a00	数据
	0xb8000	0xbad00	0x2d00	数据
PageB	0xbad00	0xbebc4	0x5300	代码数据

片外接 8 块存储器,每块大小 32 k * 16bit,片外存储器共 256 k * 16bit. 需要将 Page4/5/6/7/8/9/a/b 程序页的高 32 k 地址空间映射在 8 块存储器上,8 页程序空间需要占用 20 根地址线[A0-A19],因此对 5 根地址线[A15-A19]进行译码处理,确定选择哪一块存储器,其他 15 根地址线[A0-A14]与每块 32 k 存储器地址线相连,用于读写数据. 存储器地址映射如图 3 所示.

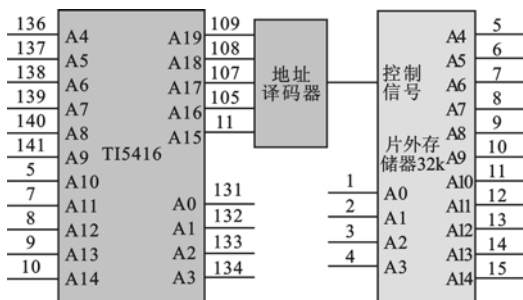


图3 存储器地址映射

上电时 MP/MC = 1、OVLY = 1、DROM = 0, 程序从 0xff80 映射的片外存储器上的第一条指令开始执行^[5],相应的空间应该存放的是 boot 程序的代码. 而 boot 程序的代码量不大,只有约 12 kbit 大小,因此将 boot 程序和语音程序的一块码本数据合并放入 page8 的高 32 k 地址空间,boot 程序放入 0x8f800 ~ 8ffff 地址空间. 将程序空间的 Page0 和 Page8 空间进行页共享,通过一个地址译码器,Page0 和 Page8 的地址空间映射在同一块的固态存储器上,执行 0xff80 即是执行 0x8f80 地址开始的程序. 这样做有效的节省了存储器容量开销. 而共享存储器如图 4 所示.

当 MP/MC = 1 时,Page0 和 Page8 页映射在同一块片外存储器上,同样的方法,将 Page9 和 Page1、Page A 和 Page2、Page B 和 Page3 分别映射在同一块

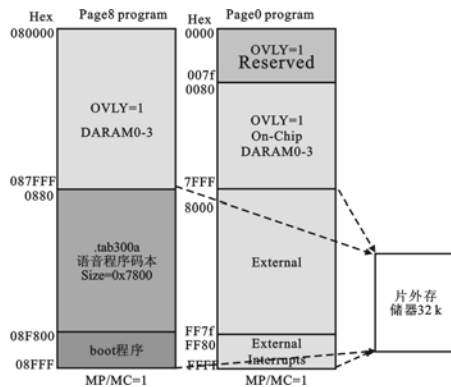


图4 页共享存储器

片外存储器上.

上电或芯片复位后,程序从 0xff80 指向的片外存储器空间开始执行,此时 PC = 0xff80, XPC = 0, 硬件上做了地址映射后,0xff80 执行的代码,即为空间 0x8f80 内的 boot 程序,boot 程序第一条指令执行跳转,跳转到 0x8f800 地址执行芯片和相关配置 (XPC = 8),然后重新进行地址映射 (PMST = 0x01a8, MP/MC = 0),接下来开始进行程序和数据搬移. boot 程序执行顺序如图 5 所示.

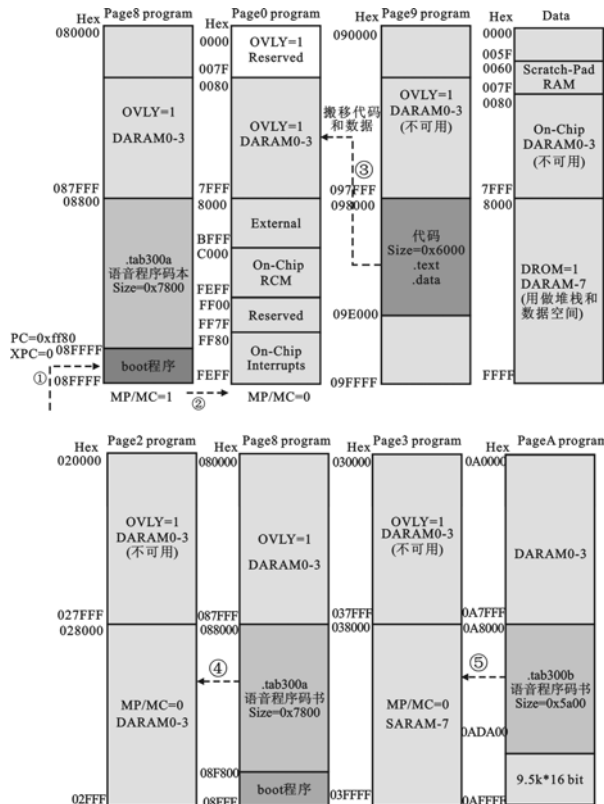


图5 boot 程序执行顺序

在搬移片外代码和数据之前,设置 $MP/MC = 0$,重新映射存储空间,再将代码和数据分别搬移到相应的片内空间,以达到最大的处理速度. 由于 TI5416 程序空间和数据空间相互独立,因此在代码搬移时,要考虑程序到程序空间,程序到数据空间的代码搬移的问题, TI5416 内部的处理器含有对程序到程序空间,程序到数据空间,数据到程序空间的数据搬移指令,但操作起来非常复杂. 而 TI5416 处理器片内含 6 通道 DMA,支持数据、I/O、程序空间的数据搬移,且搬移操作不受与地址映射有关的 MP/MC 、 $DR0M$ 、 $OVLY$ 寄存器位的影响. 因此采用 DMA 来进行代码和数据的搬移,既简单速度又快. 相应的程序和数据搬移完成后,跳转到语音程序 C 入口地址 $0x200$ ($XPC = 0$, $PC = 0x200$) 开始运行程序^[5].

3.3 程序流程

程序流程如图 6 所示. Boot 程序上电后,执行 $0xff80$ 地址指令,该指令为一条跳转指令,此时 $XPC = 0$, $PC = 0xff80$. 跳转指令将程序跳转到 $0x8f800$,此时 $XPC = 8$, $PC = 0x8f800$,执行相应的芯片配置功能后,重新进行地址映射,此时 $MP/MC = 0$,芯片工作在微计算机模式,然后配置 DMA,开始分块搬移代码和数据,搬移完成后,跳转到片内语音程序的 C 入口地址,开始运行语音程序.

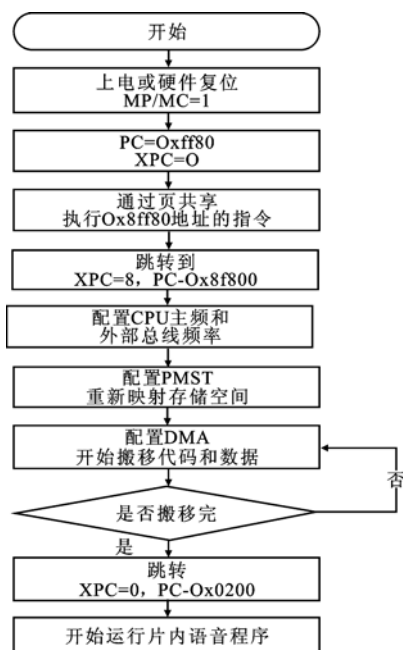


图 6 程序流程图

Boot 程序执行步骤如下:

- (1) 外部 MP/MC Pin 拉高,上电/复位芯片;
- (2) 初始化芯片和相关外围设备;
- (3) 重新映射内存空间,为应用程序准备 RAM 空间;
- (4) 启动 DMA,将片外存储器上的代码和数据分别搬移到相应的内部存储器上;
- (5) 跳转到内部语音程序的 C 入口地址,开始运行语音程序.

4 结束语

本文提出了一种基于 TI5416 的应用程序 boot 方法,利用了 TI5416 处理器用户可以不受片上 ROM 固化的 boot loader 功能的限制,可自定义 boot 程序的功能,灵活的设计了 boot 方法,成功的将复杂的语音压缩算法程序装载运行了起来. 该方法利用处理器存储结构特点,设计了一个地址译码器,有效的将 8 块外部 32 k 存储器映射到 8 页程序高地址空间,避免了反复切换地址映射引起的不必要的错误,地址译码器还实现了程序页共享机制,将 page0/8 页映射在同一块外部存储器上,上电启动时,有效节省了空间需求.

参考文献:

- [1] 郑红,吴冠. TMS320C54x DSP 应用系统设计[M]. 北京:北京航空航天大学出版社,2002.
- [2] 张雄伟,曹铁勇. DSP 芯片的原理与开发应用[M]. 2 版. 北京:电子工业出版社,2000.
- [3] 胡洪波,梁书剑,张德兴. TMS320VC5416DSP 存储空间配置方法研究[J]. 价值工程,2011(7):149-150.
- [4] 徐书杰,刘宏立. TMS320VC5416 DSP 并行自举方案的设计与实现[J]. 计算机工程与科学,2009,31(8):153-155.
- [5] 刘益成. TMS320C54xDSP 应用程序设计与开发[M]. 北京:北京航空航天大学出版社,2002.

作者简介:

张梅娟 女,(1981-),硕士,工程师. 研究方向为嵌入式软件设计. E-mail:154592485@qq.com.

石乔林 男,(1942-),硕士,高级工程师. 研究方向为数字电路设计.

俞光耀 男,(1980-),硕士,工程师. 研究方向为 DSP 软件设计.

李晓磊 男,(1983-),工程师. 研究方向为自动化控制.