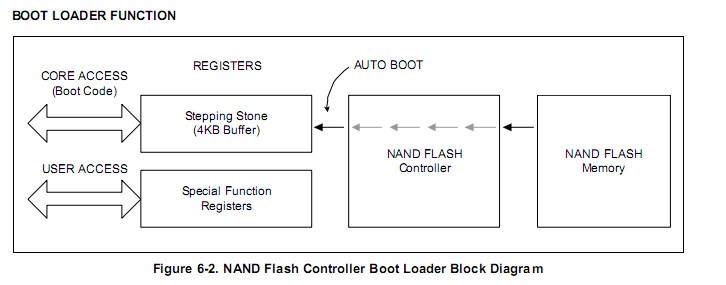
[**s3c2440存储控制器和地址以及启动的理解**](http://www.cnblogs.com/geneil/archive/2011/12/20/2295220.html)

**1.首先应该先了解Flash ROM的种类**

　　NOR FLASH地址线和数据线分开，来了地址和控制信号，数据就出来。  
　　NAND Flash地址线和数据线在一起，需要用程序来控制，才能出数据。通俗的说，只给地址不行，要先命令，再给地址，才能读到NAND的数据,在一个总线完成的。  
　　结论是：ARM无法从NAND直接启动。除非装载完程序，才能使用NAND Flash.  
　　Nand Flash的命令、地址、数据都通过I/O口发送，管脚复用，这样做做的好处是，可以明显减少NAND FLASH的管脚数目，将来如果设计者想将NAND FLASH更换为更高密度、更大容量的，也不必改动电路板。在S3C2440中NANDFLASH的控制依靠NAND FLASH控制器。不能够执行程序，总结其原因如下 ：  
（1） NAND FLASH本身是连接到了控制器上而不是系统总线上。CPU启动后是要取指令执行的，如果是SROM、NOR FLASH 等之类的，CPU 发个地址就可以取得指令并执行，NAND FLASH不行，因为NAND FLASH 是管脚复用，它有自己的一套时序，这样CPU无法取得可以执行的代码，也就不能初始化系统了。  
（2）NAND FLASH是顺序存取设备，不能够被随机访问，程序就不能够分支或跳转，这样你如何去设计程序。

**2.在2440中为什么可以配置成从Nand Flash中启动程序？**  
　　如果S3C2440被配置成从Nand Flash启动, S3C2440的Nand Flash控制器有一个特殊的功能,在S3C2440上电后,Nand Flash控制器会自动的把Nand Flash上的前4K数据搬移到4K内部SRAM中,（此内部RAM被称为Steppingstone）并把0x00000000设置内部RAM的起始地址,CPU从内部RAM的0x00000000位置开始启动。这个过程不需要程序干涉。程序员需要完成的工作,是把最核心的启动程序放在Nand Flash的前4K中，也就是说,你需要编写一个长度小于4K的引导程序,作用是将主程序拷贝到SDRAM中运行。



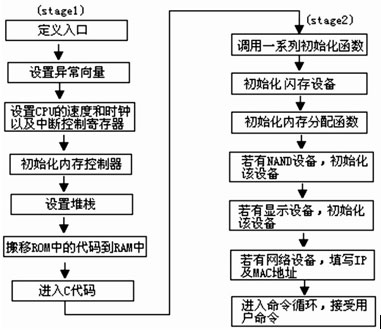
**3.启动方式：**

Samsung S3C2440支持Nor Flash和Nand Flash启动，主要由OM[1:0]这两位来决定从何处启动。具体含义如下:

OM[1：0]=00时，处理器从NAND Flash启动；  
OM[1：0]=01时，处理器从16位宽度的ROM启动；  
OM[1：0]=10时，处理器从32位宽度的ROM启动；  
OM[1：0]=11时，处理器从Test Mode启动。

　　ARM的启动都是从0地址开始，所不同的是地址的映射不一样。在ARM开电的时候，要想让ARM知道以某种方式（地址映射方式）运行，不可能通过你写的某段程序控制，因为这时候你的程序还没启动，这时候ARM会通过引脚的电平来判断。  
（1）当引脚OM0跟OM1有一个是高电平时，这时地址0会映射到外部nGCS0片选的空间(Bank0)，也就是Norflash，程序就会从Norflash中启动，ARM直接取Norflash中的指令运行。  
（2）当OM0跟OM1都为低电平，则0地址内部bootbuf（一段4k的SRAM）开始。系统上电，ARM会自动把NANDflash中的前4K内容考到bootbuf（也就是0地址），然后从0地址运行。这时NANDFlash中的前4K就是启动代码（他的功能就是初始化硬件然后在把NANDFlash中的代码复制到RAM中，再把相应的指针指向该运行的地方）

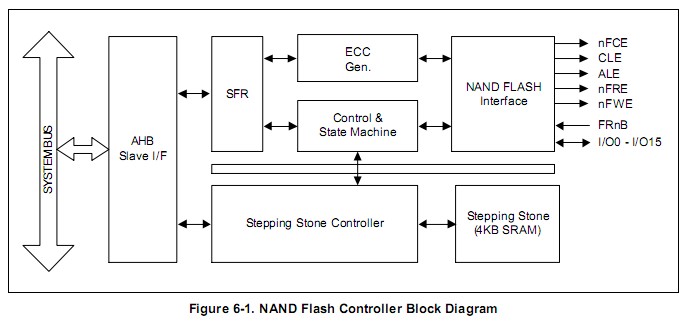
**4.启动代码应该做什么？**  
　　由于Nand Flash控制器从Nand Flash中搬移到内部RAM的代码是有限的,所以在启动代码的前4K里,我们必须完成S3C2440的核心配置以及把启动代码(U-BOOT)剩余部分搬到RAM中运行。至于将2440当做单片机玩裸跑程序的时候，就不要做这样的事情，当代码小于4K的时候，只要下到nand flash中就会被搬运到内部RAM中执行了。bootloader在某种意义上来说即是一个启动代码，种类有很多（vivi，uboot 等），但是功能上无非就是完成一些初始化。bootloader是芯片复位后进入操作系统之前执行的一段代码，完成由硬件启动到操作系统启动的过渡，为运行操作系统提供基本的运行环境，如初始化CPU、堆栈、初始化存储器系统等，其功能类似于PC机的BIOS。



　　在实际的开发中，一般可以把bootloader烧入到Norflash，程序运行可以通过串口交互，进行一定的操作，比如下载，调试。这样就很可以很方便的调试你的一些代码。Norflash中的Bootloader还可以烧录内核到Norflash等等功能。  
  
**5.存储控制器的作用：**  
　　在2440中分了8个bank,每个bank的基地址由nCGSx来选择，每个bank都接外设之后，就可以通过存储控制器来进行地址上的选择了。每个bank与外设的连接方式不一样，主要看外设是每次进行多少位的数据传输，如果是8位，这样CPU的地址线A0就可以直接接外设的A0，如果是16位，那么CPU的A1就该接到外设的A0,以此类推。nor flash接在bank0,数据线为16位。存储控制器的特性如下：  
　　大小端设置;  
　　地址空间:每个bank为128MB (总共1GB);  
　　除了bank0其余所有banks的数据位宽是可编程的(8/16/32-bit);（bank0是16/32位）  
 　　总共8个memory banks,其中6个bank是接ROM,SRAM等,其余2个bank是接ROM,SRAM,SDRAM等;  
　　7个memory bank的起始地址是固定的;(发现size也是固定的,128MB)  
　　1个memory bank的起始地址和大小是可灵活可变的;  
　　所有banks的访问周期数是可编程的;  
　　支持片外等待信号以扩充总线周期;  
　　SDRAM在Power down模式下支持自动刷新。

　　这里看到有8个BANK，然后他们地址怎么就可以确定呢？理解：外面一共就使用了27根地址线，还有5根地址线没有被使用，5根地址线就有16个选择了，初步估计其中的3根又被用来选择这8个bank了，所以才有那样的地址~~至于那个可变地址，暂时还想不出什么解释，好像是叫部分译码~呵呵。。OK

**6.Nand Flash控制器：**  
 　　自动启动: 系统复位后,boot code搬运到4KB Steppingstone,然后在其内部执行;  
 　　Nand Flash存储接口: 支持256Words,512Bytes,1KWords和2KBytes Page;  
　　软件模式: 用户能直接访问nand flash;  
　　接口: 支持8/16-bit Nand flash存储接口;  
　　支持大小端模式;  
　　硬件ECC发生器;  
　　Steppingstone: 4KB SRAM Buffer,在nand flash启动过后可以用作它处.



　　当系统处在复位状态时,Nand flash控制器从管脚NCON,GPG13,GPG14,GPG15得到nand flash的一些信息(如page size,bus width等,见下图)。在上电或系统复位之后,则Nand flash控制器将自动加载4KB boot loader代码,之后就是在steppingstone里执行.  
　　注意:在自动启动这个过程中,ECC模块是不发挥作用的。