DM365 启动分析

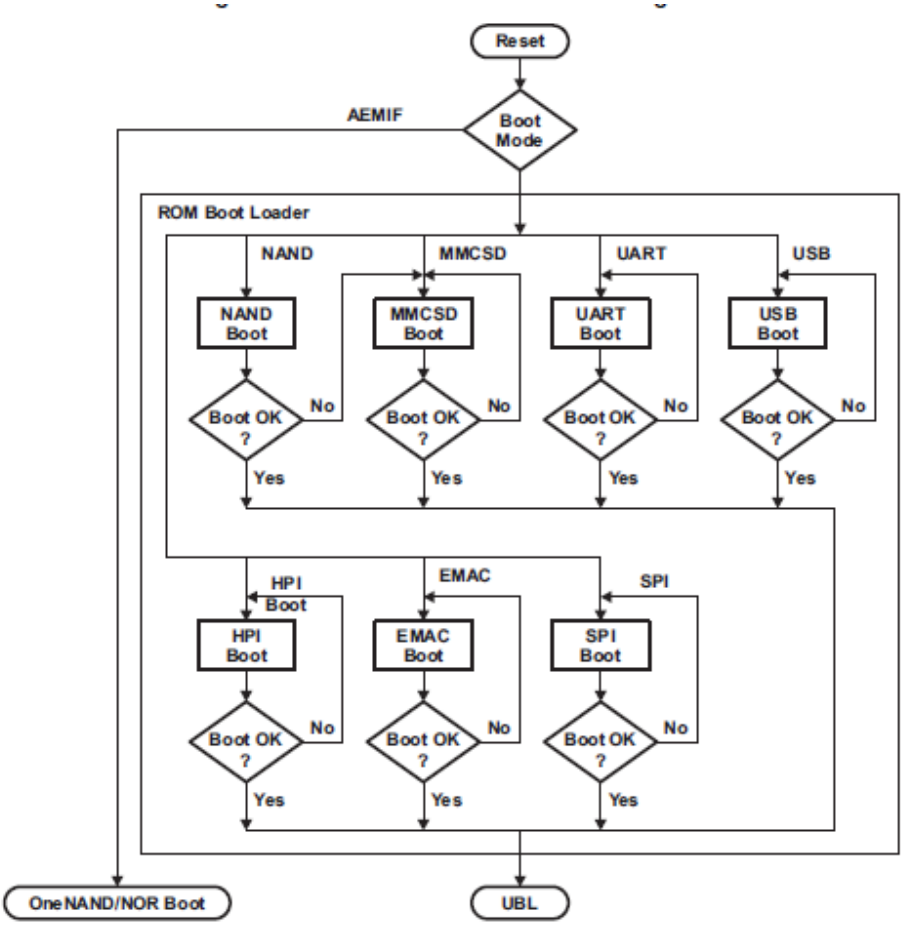
DM365 可以通过设置跳线接口来实现不同的启动模式，当硬件管脚 BTSEL[2:0]被设置成 001 时，DM365 就会通过 EMIF(NOR FLASH)来启动，而当设成其它的值时，DM365 就会通过内部的 ROM 启动。

因为 NOR FLASH 用的是 EMIF 接口，所以 NOR FLASH 非常的类似 SDRAM,如果从 NOR FLASH 启动的话，只要把正确的 U-BOOT 启动代码置于 NOR FLASH 起始位置就能够实现 DM365 的启动，具体的启动过程完全有 U-BOOT 来控制。

DAVINCI 芯片出厂时，内部 ROM 固化了一段无法擦除 BOOT LOADER，称为 RBL（ROM BOOTLOADER），它可通过 NAND FLASH 控制器或者 UART 加载一段用户的 UBL（USER BOOTLOADER）至内存的 0000:0020 地址开始执行。但 UBL 大小被限制在 14K 以内，无法达到 U-BOOT 镜象的大小，因此无法直接将 U-BOOT 作为 UBL 使 RBL 加载。所以，要提供一个小于 14K 的 UBL（狭义的 UBL），放在 NAND Flash 的前 5 个 block 内。系统启动时，先运行 RBL，RBL 加载 NAND Flash 中的 UBL，UBL 再加载放在 NAND Flash 中的 U-boot，U-boot 再加载操作系统。

RBL 运行部份

如果 BTSEL[2:0]没有被设置成 001 的话，那么 DM365 就会从内部的 ROM 启动，ARM926ES-J 复位后就会执行 ROM 固化的程序 ROM BOOTLOADER(RBL)。由 RBL 去下载 USER BOOTLOADER(UBL)并且执行 UBL。DM365 的 BOOT 过程如下：



从上图可以看 DM365 中的 RBL 具有以下特点，能够支持七个不同的引导方式，NAND BOOT,MMC/SD BOOT,UART BOOT,USB BOOT,SPI BOOT,EMAC BOOT,HPI BOOT。

NAND BOOT　　BTSEL[2:0]=000

NOR BOOT　　BTSEL[2:0]=001

MMC/SD BOOT　　BTSEL[2:0]=010

UART BOOT　　BTSEL[2:0]=011

USB BOOT　　BTSEL[2:0]=100

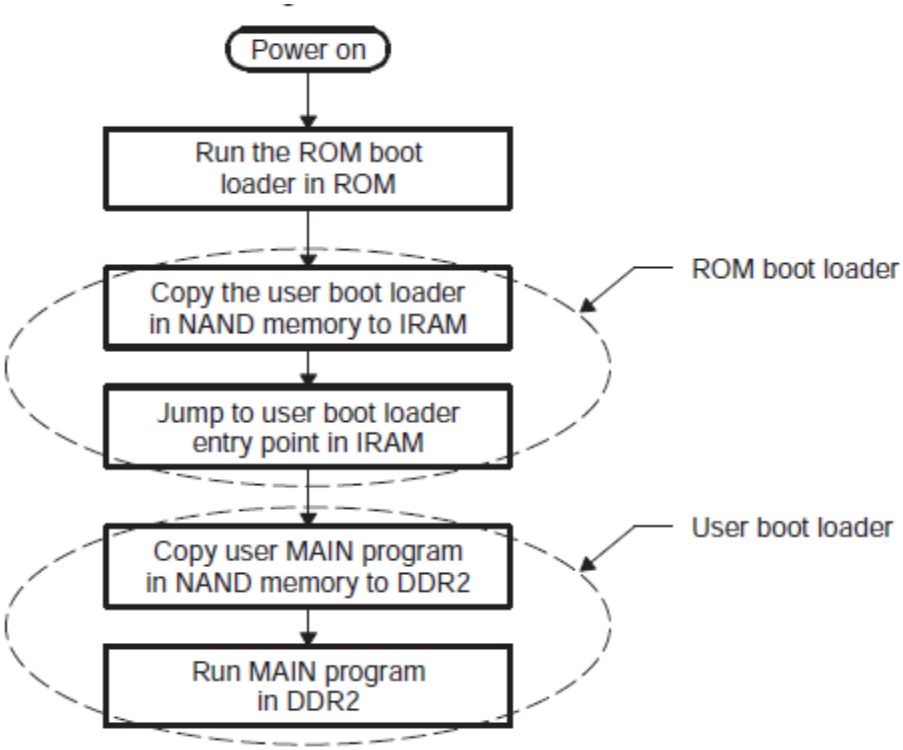
SPI BOOT　　BTSEL[2:0]=101

EMAC BOOT　　BTSEL[2:0]=110

HPE BOOT　　BTSEL[2:0]=111

在 RBL 模式下，如果 NAND BOOT 失败了，会尝试 MSC/SD BOOT,如果 MSC/SD BOOT 失败了，会接着试 MSC/SD BOOT.其它的几种 BOOT 都跟 MSC/SD 的启动过程一样，如果失败了，会接着重试。　　根据项目需求，这里主要分析 DM365 的 NAND BOOT 和 UART BOOT 过程。

如果 BTSEL[2:0]=000，NAND BOOT 将会被执行。



１）RBL 读 NAND FLASH 的设备 ID 号。

２）RBL 用读到的设备 ID 号跟自己所支持的 NAND FLASH ID 作对比，如果读到的 NAND FLASH 设备号不在所支持的范围内，NAND FLASH 启动将会失败。如果读到的 NAND FLASH 设备号是 RBL 所支持地，执行第三步。

３）RBL 将会从 NAND FLASH 中寻找 UBL 的表述信息，如果在寻找 24 个 BLOCK 后还没有找到 UBL,NAND FLASH 就会失败。如果在这期间找到了 UBL 的可用签名,执行第四步。

４）RBL 会把 UBL 存在 NAND FLASH 中的 BLOCK 数存到 ARM 内部 RAM 的最后一个字处，这个操作是为了调试的方便。

５）RBL 先读 UBL 的表述表，该表述表中存有 UBL 的一些基本信息。

６）由这个表述表 RBL 从 NAND FLASH 中把 UBL 拷贝到 ARM 内部的 RAM 0x00000020 开始处。在传输的过中 RBL 会用 4bits 的 ECC 进行较验，如果发现是其它错误所致，就会执行第三步。

其中 UBL 的表述表和签名分别如下图所示：

Page 0 Address	32-Bits	Description
0	0xA1AC EDxx	Magic number (0xA1ACEDxx)
4	Entry Point Address of UBL	Entry point address for the user bootloader (absolute address)
8	Number of pages in UBL	Number of pages (size of user bootloader in number of pages)
12	Starting Block # of UBL	Block number where user bootloader is present
16	Starting Page # of UBL	Page number where user bootloader is present
20	PLL settings -M	PLL setting -Multiplier (only valid is Magic Number indicates PLL enable)
24	PLL settings -N	PLL setting -Divider (only valid is Magic Number indicates PLL enable)
28	Fast EMIF setting	Fast EMIF settings(only valid is Magic Number indicates fast EMIF boot)

NAND UBL Descriptor

Mode	Value	Description
UBL_MAGIC_SAFE	0x A1AC ED00	Safe boot mode
UBL_MAGIC_DMA	0x A1AC ED11	DMA boot mode
UBL_MAGIC_IC	0x A1AC ED22	Instruction Cache boot mode
UBL_MAGIC_FAST	0x A1AC ED33	Fast EMIF boot mode
UBL_MAGIC_DMA_IC	0x A1AC ED44	DMA +Instruction Cache boot mode
UBL_MAGIC_DMA_IC_FAST	0x A1AC ED55	DMA +Instruction Cache+ Fast EMIF boot mode
UBL_MAGIC_PLL	0x A1AC ED66	With PLL enabled to have higher ARM/DMA clocks
UBL_MAGIC_PLL_DMA	0x A1AC ED77	With PLL enabled +DMA
UBL_MAGIC_PLL_IC	0x A1AC ED88	With PLL enabled +Instruction Cache
UBL_MAGIC_PLL_FAST	0x A1AC ED99	With PLL enabled +Fast EMIF
UBL_MAGIC_PLL_DMA_IC	0x A1AC EDAA	With PLL enabled +DMA+Instruction Cache
UBL_MAGIC_PLL_DMA_IC_FAST	0x A1AC EDBB	With PLL enabled +DMA+Instruction Cache+ Fast EMIF
UBL_MAGIC_SAFE_LEGACY	0xA1ACEDCC	Safe boot mode with legacy

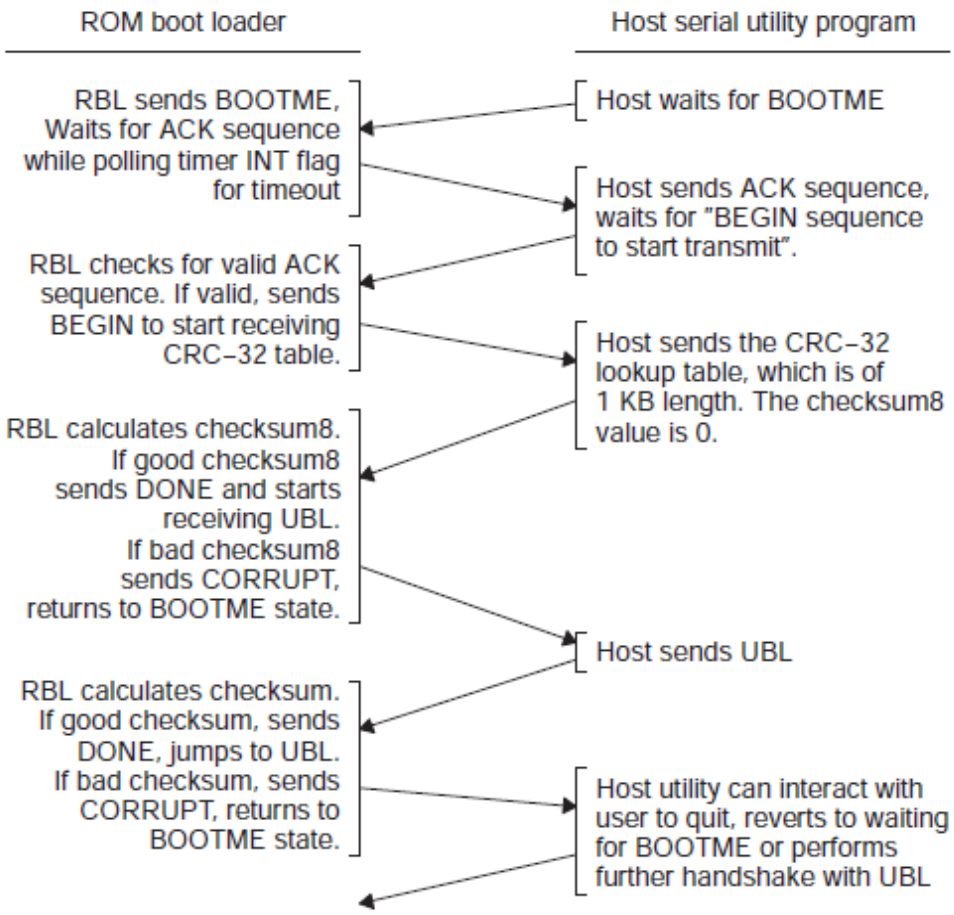
UBL Signatures and Special Modes

如果 BTSEL[2:0]=011，UART BOOT 将会被执行。

UART BOOT 方式将会涉及到一个主机的程序，RBL 通过与主机程序之间的交互来完成 UBL 的传送过程。串口的设定如下：

Time-Out	500 ms, one-shot
Serial RS-232 port	115.2 Kbps, 8-bit, no parity, one stop bit
Command, data, and checksum format	Everything sent from the host to the DM36x UART RBL must be in ASCII format.

RBL 与 Host Program 的通信过程如下：



1）RBL 发送 BOOTME,然后等待 ACK,直到超时

2）Host 接收到 BOOTME 后，发送 ACK,然后等待 BEGIN

3）RBL 接收到 ACK 后，检查其合法性，如果合法可用，发送 BEGIN,不合法就接着等待

4）Host 接收到 BEGIN 后发送 CRC-32table

5）RBL 接收到 CRC-32table 后检查 checksum 值，如果正确发送 Done，如果不正确发送 CORRUPT,然后回到第一步

6）Host 接收到 Done 后开始 UBL

7）RBL 接收 UBL 并计算 checksum 值，如果正确发送 Done，如果不正确发送 CORRPUT,然后回到第一步。

UBL 运行部份

UBL 分跟 U-BOOT 一样，分为发布 UBL 和测试 UBL。

发布的 UBL 主要有以下功能

- 将从 NAND Flash 上将 U-boot 加载到内存并执行。

- 当 UBL 发现在 nand flash 上不能读取到 U-boot 时，将会打印 BOOTPSP，此时可以通过串口发送 U-boot（该功能目前未提供）或者等待一段时间后（打印 10 个 BOOTPSP）ubl 将自动将 NAND Flash block 1～5 擦除，并自动跳转到 RBL 执行，此时串口将打印 BOOTME。

测试的 UBL 主要完成真正 UBL 的下载与烧录

- 通过 UART 接口接收正式 UBL（.bin 格式）和 u-boot.bin 文件
- 自动把 UBL 烧录到 Block 1～5 上，把 U-boot 烧录到之后的 Block 上。

在这种情况下，测试 UBL 首先被 RBL 由 UART 读入内存，然后把控制权交由测试的 UBL,而测试的 UBL 再通过 UART 口接收正式的 UBL 和 U-BOOT 然后把它们烧录到 NAND FLASH 中。

U-BOOT 部份

RBL 把控制权交由 UBL 后，UBL 会从 NAND FLASH 中拷贝出 U-BOOT,然后把控制权交由 U-BOOT。U-BOOT 完成启动，然后调用 LINUX KERNEL,并把控制权交由 KERNEL。

DM365 的 u-boot 启动过程跟其它的 ARM 启动过程类似，它的启动如下所示：

LINUX KERNEL 启动部份

当 u-boot 把控制权交由 LINUX KERNEL 时，u-boot 已经完成以下任务：

1.CPU 必须处于 SVC(supervisor)模式,并且 IRQ 和 FIQ 中断都是禁止 的;

2.MMU(内存管理单元)必须是关闭的,此时虚拟地址就是物理地址;

3.数据 cache(Data cache)必须是关闭的

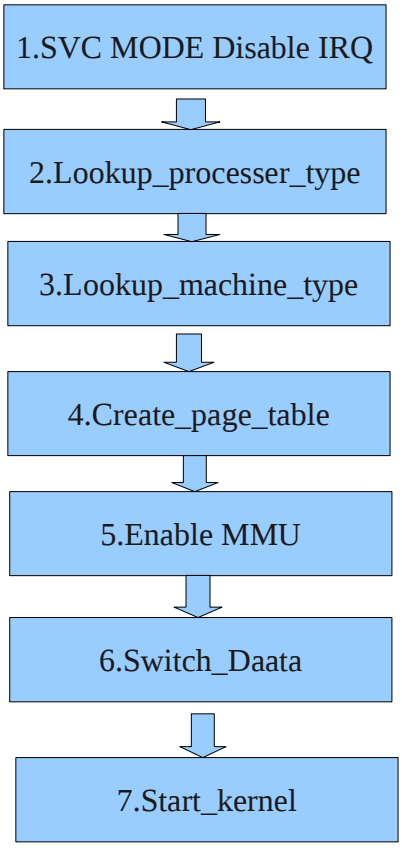
4.指令 cache(Instruction cache)可以是打开的,也可以是关闭的,这个没有强制要求;

5.CPU 通用寄存器 0(r0)必须是 0;

6.CPU 通用寄存器 1(r1)必须是 ARM Linux machine type(关于 machine type, 我们后面会有讲解)

7.CPU 通用寄存器 2 (r2) 必须是 kernel parameter list 的物理地址 (parameter list 是由 boot loader 传递给 kernel,用来描述设备信息属性的列表,详细内容可参考"Booting ARM Linux"文档).

DM365 在把内核进行自解压后，会从 arch/arm/kernel/head.S 从文件处执行。



1.出于谨慎的角度，LINUX KERNEL 对 ARM926EJ-S 又进行了一次 SVC MODE,Disable IRQ 设置。

２.通过 CP15 获得 cpu id 号，通过 Cpu id 号来查询 proc_info_list 得到一个结构体指针 proc_info。

３.由 u-boot 传过来的 machine_type 在 arch_info 表中查找到 machine_desc 结构指针。

4.由 machine_desc 和 proc_info 里面的信息，创建内存映射表，并且初始化内核的映射表。

5.由 machine_desc 和 proc_info 里面的信息，打开 MMU

6.进行一些数据的必要赋值

7.调用 start_kernel

在 LINUX KERNEL 调用 start_kernel 后，会分别初始化跟 DM365 体系相关的 map_io_init,irq_init,timer_init,machine_init.

zhaoxin@kedacom.com

2010.07.14