1. 概述

Bootloader就是嵌入式系统上电后执行的第一段代码（包括厂商固化的一级Boot和各种第三方的boot程序，如uboot），它负责初始化CPU以及相关的硬件（如DDR，串口等等），将操作系统镜像或者应用程序加载到RAM中。

1. 工作模式
   1. 启动模式

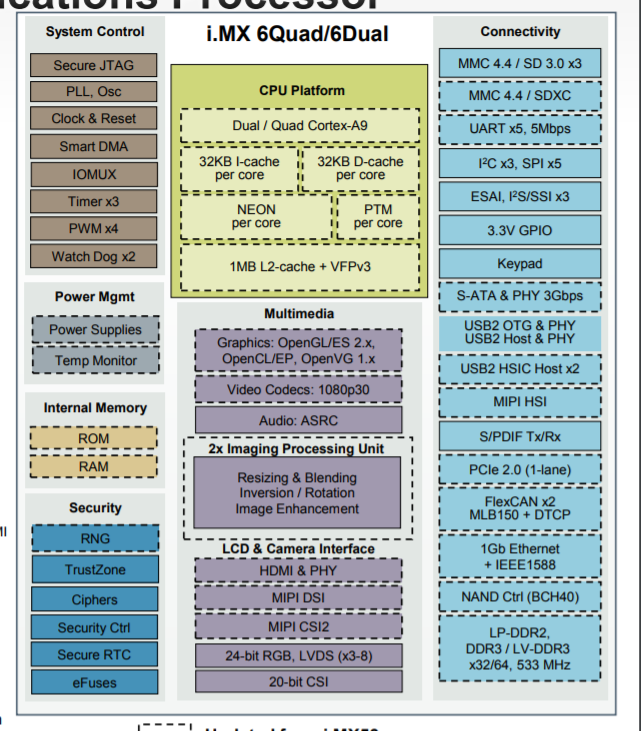
加载模型也叫自启动模式，即bootloader从某种存储介质中读取操作系统镜像，并跳转到对应地址执行。

* 1. 下载模式

通过串口或者网口，将操作系统镜像或者应用下载到RAM中，再跳转到RAM中运行操作系统或者再将RAM中的操作系统烧写到某种掉电不丢失的存储介质中。

1. 启动实例分析
   1. Freescale IMX6q处理器
      1. 简介

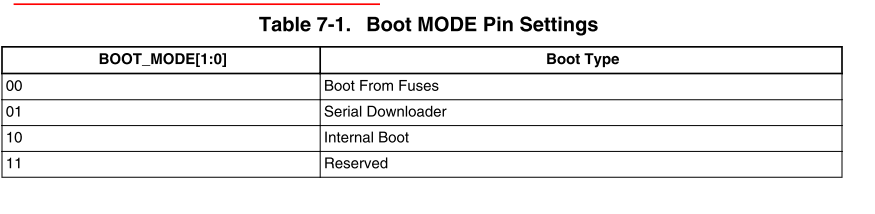
IMX6Q处理器是基于Cortex-A9的4核高性能ARM SOC，其SOC的抽象框图如下：



1. IMX6Q
   * 1. 启动模式

Imx6q上电后，会先执行内部（IROM）代码，这段代码的主要作用就是配置启动模式。

IROM代码支持4种启动模式，11为freescale保留的。

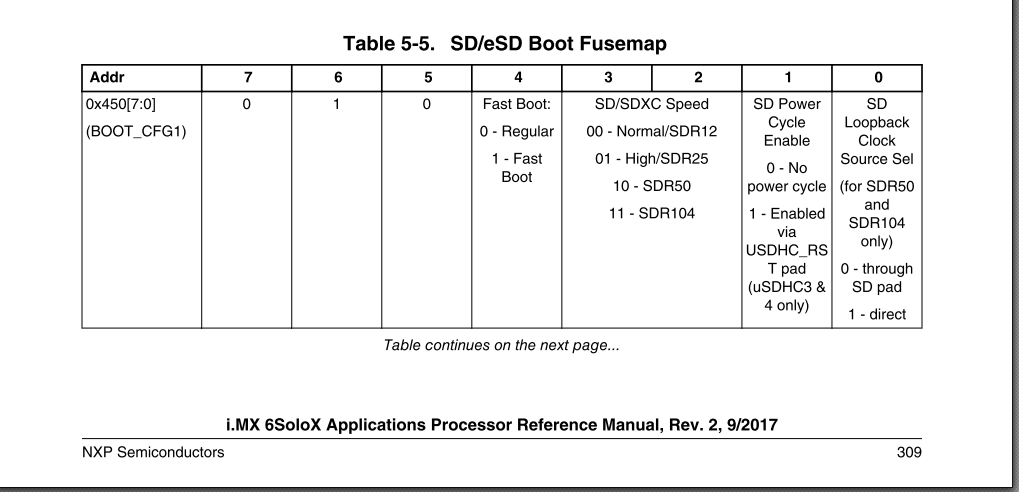


**Boot From Fuses:**

先解释下什么是IMX6如何决定boot模式。通常情况下，处理器都会对外提供BOOT的配置引脚，以便处理器的固件根据不同的配置来决定从哪个地方启动。IMX6也不例外，它提供了若干个BOOT配置引脚共我们使用，但是，IMX6还提供了一种特殊的BOOT模式——boot from fuses。

boot from fuses和读取GPIO引脚来决定启动方式的区别在哪里呢？看官方手册的说明：The Boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio\_bt\_sel e-fuse defines the source to be used to derive the bootinformation. When gpio\_bt\_sel is set, e-fuses are used. When cleared, GPIO signals areused.

简单说，就是IMX6的可以支持读取e-fuses(固化在OCOTP-onchip one-time programmable)寄存器中的值来进行BOOT启动方式配置。而OCOTP中的fuse值与BOOT 配置的GPIO也存在着映射关系，如下图：



1. Fuse map

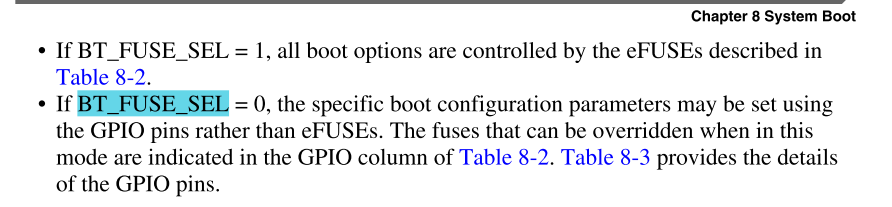
从上表可以看出，偏移地址为0x450的寄存器（OCOTP）对应了BOOT\_CFG1的8根引脚。

实际上，e-fuses可以形象得理解成一次性可编程的保险丝，它的本质是寄存器，只不过该寄存器写过一次，再上电复位后（换言之，没有上电复位前可以任意写）就不能再写。它的好处是就是当我们有量产的IMX6时，就可以把所有的板卡配置成这种启动模式，然后只需要将他们的eFuse烧写成统一的值，那么当这些板子上电复位后，就统一的从某个存储介质上启动，而不需要再像传统的方式一样：

切换到下载模式-----下载镜像-----切换到启动模式

**Serial Downloader:**串口下载模式。通过USB线将PC和imx6相连，在设置imx的USB为DEVICE模式，PC端通过MFGtool就能实现代码固化。（固化后需要启动的话，将启动模式设置为对应模式即可）

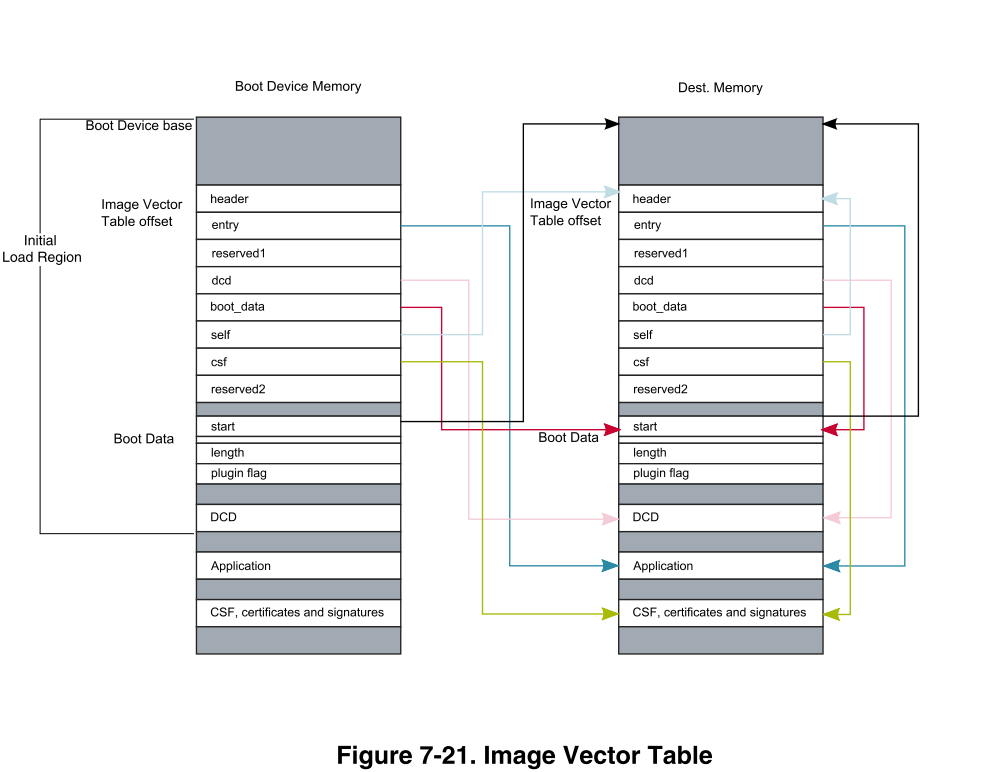
**Internal Boot:**和该模式下，CPU会继续执行IROM的代码，并根据BT\_FUSE\_SEL的值确定是否使用GPIO（CFG\_PIN）的值来覆盖部分eFuse（此时是外部引脚控制）的值。



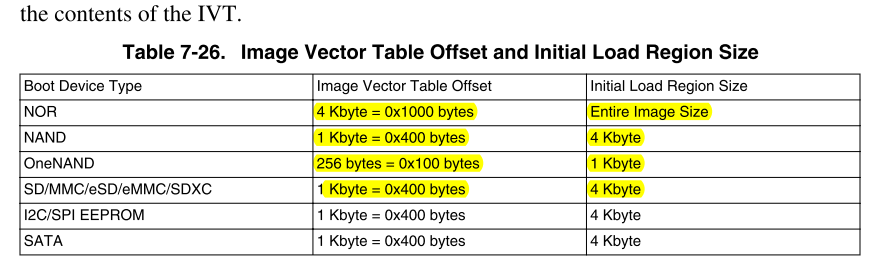
默认情况下，BT\_FUSE\_SEL为0，也就是根据外面的GPIO管脚来决定启动位置。这种模式和Boot From Fuses最大的区别就是不会强制进入Serial Downloader模式。

* + 1. 启动uboot流程

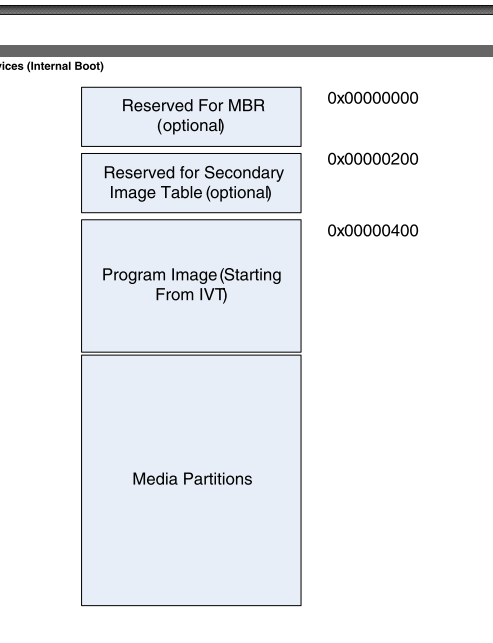
Imx6的启动在其RM手册中有详细说明，核心是CPU上电后会执行内部IROM中的固件代码，这段代码的主要作用是检查imx6的boot模式和boot device。以我的应用为例，Boot模式为internalROM和emmc，那么，当我们把uboot烧写到emmc中后，imx6会一次性读取emmc的前4K字节的数据，这段数据主要是配置DDR等系统模块，配置完后就将uboot整个拷贝到DDR中，跳转到uboot的入口函数\_start函数开始执行。（为什么IROM中的代码知道跳转到uboot的入口，其奥秘就在IVT中，IVT提供了entry的参数，这个参数的值就是uboot入口函数\_start的地址）



事实上，imx6的uboot.bin就是以IVT+uboot本体组成的，IVT就是配置数据，不同的boot device IVT的存储位置有所差别，如下图：



从上表可以看出，我的应用（emmc）场景中，IVT存储在emmc开始偏移1K的位置，也就是如下图所示：



IVT更多的内容在IMX6的RM手册中有详细说明，重点是在Uboot生成imx6的IVT数据主要有两种方式：

uboot2015以前：通过flash\_header.S来生成IVT头；

uboot2015以后：通过mkimage –T imximage选项来生成IVT的头