**8926平台boot过程分析**

高通8926平台启动过程分析

     　作者： 张祥   　版本： 1.0

修改记录     　修改人 日期 修改内容   　张祥 2014/3/15 创建文档

1. 前言

近期公司会上8926平台项目，领导分配让我先熟悉其启动过程，起初看到代码很是傻眼，跟我之前熟悉的8x25平台是天壤之别，起初硬着头皮分析代码，看了些天发现还是一头雾水，被8x25的架构给套住了，总是以8x25平台的原理去想8926平台的问题，最后一点点分析高通文档，再结合代码才最终搞清楚了8926平台软件的大致架构．由于能力有限，加上写作功底欠佳，其中难免会有错误之处，恳请指正，如有误导大家之处，恳请理解． 本文档课题很大，在这短短的篇幅里确实介绍的不够详细，很多地方也只能意会，不好言传，需要亲自去阅读代码才能有更好的理解．这篇文档是从驱动的角度去写的，对于应用的同事仅供欣赏．总之一句话，不管你能不能看懂，反正我是懂了。

1. 名词解释：

   以下是本文档中会出现的各种英文缩写：

SMEM : shared memory

RPC : remote procedure call

PBL : primary bootloader

QCSBL  : qualcomm second bootloader

OEMSBL  : oem second bootloader

AMSS    : Advanced Mobile Subscriber Software

RPM : Resource Power Manager

SDI :System Debug Image

QSEE :  Qualcomm Secure Execution Environment

TZBSP  :  TrustZone BSP

1. 8x25平台启动过程

这篇文档主要是分析8926平台的启动流程，通过阅读这篇文档你可以大概了解8926平台是如何运行起来的．   8926平台架构与之前熟悉的7x27,8x25系列的架构完全不同，与8974平台很类似，很多代码都是共用的8974平台的代码．

 为了方便对比，这里先简单介绍一下8x25平台的启动流程，从总体架构来说，以前的8x25平台主要是分为modem端和ap端，modem端主要是处理一些射频/audio/电源管理等，系统上电后先是从高通内置ROM中的PBL程序开始启动,（这里的**PBL是不开源的**，我们不知道里面的内容也不需要修改它，我们**常用的emmc烧录软件的方法就是通过这里的代码来实现的**），然后直接进入modem端的bootloader中运行（即qcsbl和oemsbl），然后由modem的oemsbl来启动modem的操作系统amss,这里的**amss系统本质上跟linux操作系统一样，它可以执行多任务，而且有它自己的进程间通信机制**，amss操作系统会启动很多个任务（曾经统计过大概有100多个）,常见的任务有sleep\_task,snd\_task,hs\_task等，所以解跟modem端相关的bug的话可以从这些task为切入点去看代码，而不需要去看qcsbl和oemsbl，甚至根本不需要去了解amss操作系统的原理，就好比我们调试外设驱动的时候根本不需要深入分析linux内核一样．例如耳机插拔事件检测和POWER/END键功能都是通过hs\_task任务去实现的，如果耳机挺拔功能出现异常，就需要分析hs\_task的代码，搞清楚它的流程，这样就可以最终解决问题．上面的过程概要的介绍了modem的启动，可是ap端是在何时启动的还不知道，其实ap端是在modem的bootloader里跟modem内核一起启动的，在modem的bootloader中会加载appsbl（即ap端的bootloader lk）镜像文件到内存中,然后会通过设置一个寄存器来启动ap端的arm处理器．这样的话两个处理器就在同时运行．这两个处理器之间是通过一些机制来进行通信的，常见的有SMEM(共享内存)和RPC机制，上传数据比较少的话或者是只上传一个事件一般都是用RPC机制，比如耳机插入事件，modem端的hs\_task检测到耳机插入事件后会通过rpc机制向AP端上报事件，再由AP端接收事件并通过标准的input机制上报给层．

1. 8926平台启动过程

  以上简单介绍了8x25平台的启动流程，更早的7x27平台跟它很类似，应该是一个系列．可是现在的8926平台则完全颠覆了上面的启动流程．通过近段时间的学习，对其有了稍许理解，在此分享给大家，由于水平所限，其中肯定会有误解之处，欢迎大家的指正。

 先概要地说下8926平台，从硬件上说它不像8x25平台只有modem和ap两个处理器，8926平台有多个处理器，其启动流程也完全不一样，在8x25平台modem端还有许多地方需要我们自己去调试，比如audio．在8926平台上我们需要修改的地方应该会非常有限，甚至说根本就不需要修改，其几乎所有功能都放在了ap端，modem端只是处理射频相关模块．

   下面的截图是从高通文档中获取的，从中我们可以知道，8926平台是由多个处理器构成的．

     下面这幅截图也是从高通文档中获取的，这幅图介绍了8926平台的启动过程，（PS : 很多时候看高通文档能让我们更迅速的掌握高通平台原理，不看文档的话很多地方你光靠分析代码是无法搞清楚的），我们就以下面这幅图为突破点来分析8926平台的启动过程：

       系统上电后，第一个执行的CPU是Cortex-M3,其运行的是RPM镜像的PBL软件,这个RPM 的PBL程序和后面的APPS PBL,Modem PBL应该都是固化在ROM中的，高通不对我们开源，至少我没有在我们的软件中找到这三个PBL对应的代码．通过上面的图就很明朗地看出8926平台的启动过程了．过程如下：

1. 机器上电后会首先执行RPM PBL（不开源）
2. 进入APPS PBL中去执行（不开源）
3. 执行SBL1镜像文件，代码位于”/boot\_images/core/boot/secboot3/hw/msm8974/sbl1/sbl1.s”,这里是一个启动过程中非常重要的地方，分析代码你会发现它主要功能便是启动TZBSP,SDI,RPM,Appsbl四个模块．

 4.5.6.这三步是通过SBL1来启动TZBSP,SDI,RPM三个模块，它们都是8926区别于8x25平台所特有的功能．它们的代码分别位于＂/trustzone\_images,/debug\_image,/rpm\_proc＂目录中．其功能介绍见附录1.

7.RPM PBL也可以调用RPM功能，RPM模块相关的代码位于＂/rpm\_proc＂目录中．

 8.通过UEFI接口调用Appsbl，Appsbl就是我们熟悉的**lk**，它是用来启动linux的bootloader.

（PS:通过百度可以知道这里UEFI，全称“统一的可扩展固件接口”(Unified Extensible Firmware Interface)， 是一种详细描述类型接口的标准。这种接口用于操作系统自动从预启动的操作环境，加载到一种操作系统上。）

9.由Appsbl启动HLOS，即通过lk启动linux，这里的HLOS是HighLayerOS,即高级操作系统的意思，这里指linux. n,n+1,n+2,n+3,n+4.由linux来启动modem，wifi，adsp，sensor，venus等模块．可见modem是在这个时候被ap端来启动的．

（PS : 这里的venus模块暂时还不知道是做什么的）

附 录   1.8926平台的软件代码结构

目前我们8926平台的软件跟8x25一样，都是通过svn管理的，公司大多数人都只要从服务器上下载kernel和 部分的代码，其编译环境我们也很熟悉，是通过ubuntu下的类似arm-linux-gcc的工具进行编译，代码是通过ubuntu下java环境编译．而我们还有一部分代码是放在服务器上的，在8x25平台上，这部分代码统称为modem端代码，而在8926平台上因为有很多个处理器，处理modem的处理器只是其中一个，所以并不能称这些代码为modem端代码，但是出于习惯我们暂且还称它为modem端代码，这些文件的源码基本都是通过在window系统下搭建的编译环境来编译，

（PS:这些代码应该是不能在ubuntu等基于linux的操作系统中编译）．

  下面是服务器上的modem端代码结构图：

 上图中差不多每个目录都会编译成一个镜像文件，所以每个目录基本都是独立的一套程序，互相之间没有函数调用．下面逐个对上面的文档进行简要介绍．

（PS : 这些软件代码基本不需要进行深入分析，因为这些代码基本都是成熟的功能块，不需要修改，我自己也没有深入分析过，只是了解它们的大概功能，如果以后碰到问题需要分析某块代码，可以通过链接脚本文件来定位到第一个执行的文件，链接脚本文件在linux的gcc编译系统下是后缀名为”.ld”的文件，而在高通modem端的编译环境里链接脚本的后缀名为”.scl”，可以在目录下搜索这个后缀的文件，每个镜像文件会对应一个scl文件，这个scl链接脚本文件主要包含了镜像文件的存储信息，我们通过它可以断定镜像的第一个文件，基本上这第一个文件都是汇编文件，然后以这个文件为线索去分析整个软件系统架构）

   adsp\_proc : 这个目录保存的是adsp处理器的软件代码，adsp是与audio相关的dsp,audio的音频数据就是通过这个处理器处理的．值得注意的是，很多传感器的驱动也在这里面，8926平台的很多sensor的驱动软件跟8x25平台完全不一样，它们的驱动文件不是在linux中，也不是通过input机制上报数据．有关8926平台下sensor的机制我目前不是很清楚，其软件代码是在这个目录下．

   boot\_image : 顾名思义，这里存放的是与启动相关的代码，前文中提到的SBL1的代码就位于这个目录下，具体位置为：/boot\_images/core/boot/secboot3/hw/msm8974/sbl1/sbl1.s,分析启动过程时，这是一个很重要的文件，很多功能是通过它来开启的，比如trustzone,rpm,sdi，最后也是通过它来加载lk,由lk加载linux,继而最终加载的．需要注意的是，这个代码已经不属于modem端代码了，它是运行在ap处理器上的．所以说8926平台是先启动ap端后由ap端启动modem端．

（PS：其实这里的modem端与8x25平台的modem端已经不是一个概念了，这里的modem只是单纯的处理手机通话的功能）

  common : 这个目录里面存放了主要的编译工具和一些trace32调试工具的脚本文件．

 debug\_image : 这个文件夹下的文件最终会编译成一个简称为**SDI**的镜像文件，该镜像文件的主要功能是在系统崩溃后会通过安全看门狗主动把当时的程序执行环境dump到存储器中，这样有利于解决死机问题．这是8926平台区别于8x25平台的一项新的功能，这项功能可以不打开．详细介绍可参考高通文档 : 80-NA157-16\_B\_System\_Debug\_Images.pdf

   LINUX : 高通提供给客户的软件代码中，linux和软件源码是放在这个文件夹下的，但我们公司出于方便，他它单独拉出来放在ubuntu环境下进行编译．

   make : 这个目录主要存放modem端的编译程序，通过它可以编译所有需要的镜像文件，需要注意的是，并不是所有编译脚本文件都在这个目录中，它能编译所有镜像文件是因为它会依次调用其它目录下的编译脚本．我们建项目时需要对这里面的文件进行修改．

   modem \_proc : 这个目录下的文件都是与手机通信相关的代码，即真正的modem端代码．与8x25平台的modem端代码概念已经不一样了，8x25平台的modem端是包括了很多功能，比如audio,但是8926平台的modem端只与通信模块相关，有关射频的代码我也没有分析过，对于**驱动工程师**来说似乎也不太需要分析，基本不会修改到其中的代码．

  rpm\_proc : RPM是一个单独的处理器,其目标是降低MSM的平均功耗.其有三种工作模式 :power-efficient  flexible  extensible．其主要功能是动态的进行**电源管理**．具体的管理电源的几种方式可参考高通文档或直接分析源码．解决电源管理方面的问题或许要看这里的码．

   trustzone\_images : TZ是一个基于硬件的安全环境, 是Krait处理器(ARM7系列)的安全模式,类似于Supervisor模式.　TZ软件层由两个主要部分组成 : TZBSP 和 TZOS/QSEE

 TZOS/QSEE提供两个主要功能: 1. 系统软件和硬件在从下电状态到启动和唤醒期间初始化系统安全环境.

 2. 在系统运行期间为存储空间和其它子系统提供保护和服务.(这里可能翻译不太准确,英文原文见高通文档)

TZ是在SBL启动阶段被加载,用户可以自定义和编译TZ镜像文件,包括TZBSP和TZ/QSEE(tz.mbn\tzbsp\_no\_xpu)

  wcnss\_proc : WCNSS是高通的wifi模块，这个目录里主要存放了与wifi相关的固件．