**版本信息**

|  |  |  |
| --- | --- | --- |
| **文件状态： [√] 草稿 [ ] 正在修改**  **[ ] 正式发布** | **部门** | **系统软件组** |
| **版本** | 0.1 |
| **作者** | **朱坤华** |
| **完成时间** |  |
| **审核** |  |
| **审核时间** |  |
| **密级状态：绝密( ) 秘密( ) 内部资料(√) 公开( )** | | |

**修改记录:**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修订者 | 时间 | 说明 |
| 0.1 | 朱坤华 | 2018/03/16 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**一. 背景及问题:**

当同个芯片下项目众多，不同的项目对应的硬件产品的配置可能会不相同，比如项目之间可能DDR、LCD有差异，当有几个这种配置时，不仅我们开发的时候会容易混淆，客户也容易混淆。当客户烧录软件后出现问题事，由于客户可能不清楚之间差异，这可能会需要我们花不少时间弄清楚客户手上的板子是什么配置。如何防止大家都不会弄错呢，版本信息就是用于解决这个问题。

首先，一个配置的板子对应一个dts文件，不同配置的板子对应不同的dts文件。

然后，我们会预先在板子上烧录一个版本信息文件，文件的内容就是板子的配置，接着我们将多个dts文件打包到一个dtb.img里面，并烧录到板子上。

开机的时候，板子上的版本信息文件内容跟dtb.img里面的多个dtb文件一一匹配，匹配成功则加载dtb，不成功可能会导致不能成功开机。如此，我们完美的解决了问题，不管客户手上的板子是什么配置，我们总能用一个dtb.img就搞定，因为它能自动匹配对dtb。

但版本信息在代码中具体是如何跟dts匹配的呢？如何在其他代码中移植应用这个版本信息？

**二. 思路和方法:**

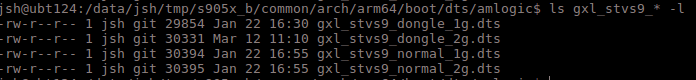
问题来了，我们先想想：

1. 如何把多个dts做到一个dtb.img里面？
2. 版本信息烧录到板子什么地方去了？
3. 如何再从dtb.img里面读到dtb？
4. 如何匹配版本信息？

好了，我们一个一个问题顺序了解：

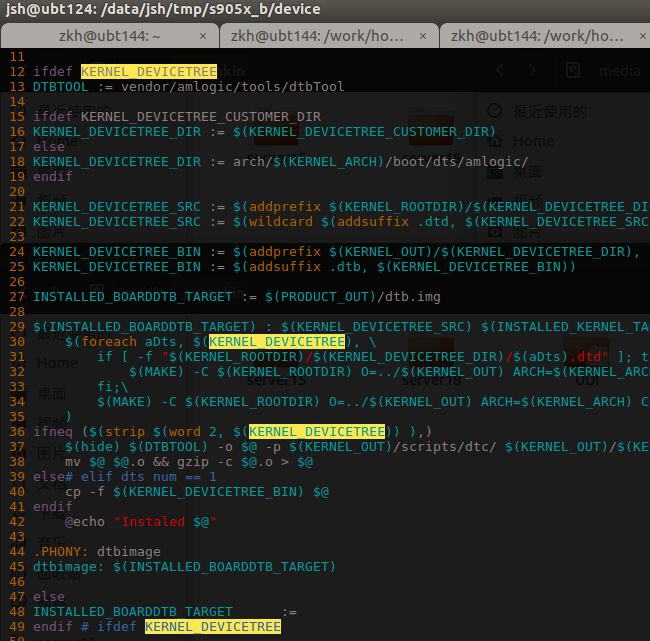
**问题一：如何把多个dts做到一个dtb.img里面？**

以S905X为例，因为这个项目下面已经有好几个配置了。如图：



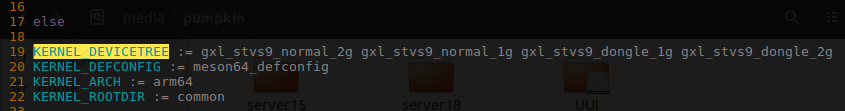
这几个文件我们是想要打包到dtb.img里面去的。如何打包呢？

Amlogic编译dtb.img的方法是执行完环境和lunch完产品后，直接make dtbimage就可以编译，追踪一下编译规则，我们可以找到编译dtb的mk文件是：device/amlogic/common/factory.mk，编译规则：

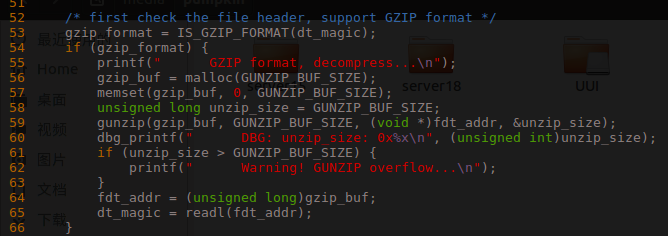


这里面有两种情况：第一种是用将单个dts编译成dtb文件，然后直接把dtb拷贝成镜像文件。第二种，假如我们编译多个dtb文件的时候，也就是支持多个dts，这时会将这几个dtb文件最终用gzip压缩到dtb.img里面。

其中编译了哪几个dts文件，由KERNEL\_DEVICETREE决定，我们再找下编译的mk文件可以找到：device/amlogic/stvs9/Kernel.mk，这时我们可以把我们想要编译的dts加进去。



前面说到压缩dtb到dtb.img，这个文件烧录到板子上后，uboot将会读出来做一个判断，用于区分gzip格式的镜像和dtb拷贝的镜像，uboot/common/aml\_dt.c



如果是gzip格式的文件，会解压后读取里面的内容，若不是gzip格式则就会直接返回dtb的地址。

所以，多个dts会用gzip 把编译出来的多个dtb文件，压缩到dtb.img里面去。

dtb编译出来后，我们会讲dtb烧录到指定的地址上去：

fatload usb 0 1080000 dtb.img;

store dtb write 1080000;

**问题二、如何匹配版本信息？**

首先，我们来看看dtb的数据存储结构，因为我们下面会用到里面的头信息。

单个dtb的dtb.img的来源是直接将后缀dtb的文件拷贝成img后缀的文件，其数据存储方式：

|  |
| --- |
| DTB header |
| alignment gap |
| memory reserve map |
| alignment gap |
| device-tree structure |
| alignment gap |
| device-tree string |

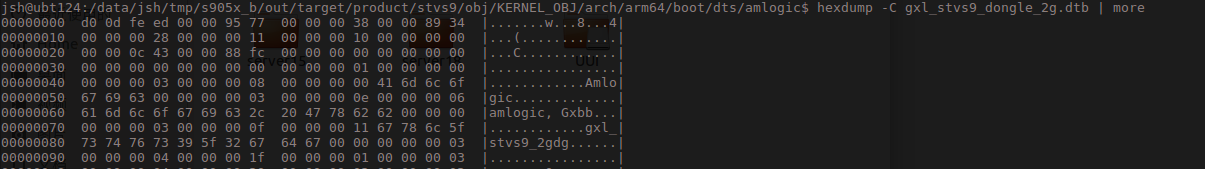
dtb header结构：

|  |
| --- |
| magic |
| totalsize |
| off\_dt\_struct |
| off\_dt\_strings |
| off\_mem\_rsvmap |
| version |
| … |

再看看多个dtb的 Dtb.img 结构

|  |  |  |
| --- | --- | --- |
| 数据 | 偏移 | 值 |
| 头 | 第0~3字节 | 0x5f4c4d41 |
| 版本控制 | 第4~7字节 |  |
| dtb数量 | 第8～11字节 |  |
| 第一个dtb数据 | 起始12～ |  |
| 第二个dtb数据 | ... |  |
| ... | ... |  |
| 第N个dtb数据 |  |  |

其中，单个dtb做的dtb.img的dtb 头的magic是一个固定的值，0xd00dfeed（大端）或者0xedfe0dd0（小端）。用命令可以直接看到：

 可以看 到dtb的前面4个字节就是**0xd00dfeed**，也就是magic。综上，我们只要提取待验证dtb的地址上的数据的前四个字节，与**0xd00dfeed**（大端）或者**0xedfe0dd0**（小端）进行比较，如果匹配的话，就说明对应待验证dtb就是一个合法的dtb，代码common/aml\_dt.c里面判断如果是单个dtb的dtb.img，会直接把fdt的地址返回回去。

**三. 技术总结**

主要描述此问题相关技术总结, 可以有哪些扩展的应用, 可以用来解决哪些其它的问题.

文档整体描述简洁明了, 可以插入代码及图片以助理解.