# 一、什么是设备树

1、uboot启动内核用到zImage，imx6ull-alientek-emmc.dtb。bootz 80800000 – 83000000.

2、设备树：设备和树。

3，在单片机驱动里面比如W25QXX，SPI，速度都是在.c文件里面写死。板级信息都写到.c里面，导致linux内核臃肿。因此 将板子信息做成独立的格式，文件扩展名为.dts。一个平台或者机器对应一个.dts。

# 二、DTS、DTB和DTC的关系

.dts相当于.c，就是DTS源码文件。

DTC工具相当于gcc编译器，将.dts编译成.dtb。

.dtb相当于bin文件，或可执行文件。

通过make dtbs编译所有的dts文件。如果要编译指定的dtbs

make imx6ull-alientek-emmc.dtb

# 三、DTS基本语法

1、设备树也有头文件，扩展名为.dtsi。可以将一款SOC他的其他所有设备/平台的共有的信息提出来，作为一个通用的.dtsi文件。

1、DTS也是’/’开始

2，从/根节点开始描述设备信息

3，在/根节点外有一些&cpu0这样的语句是“追加“。

4，节点名字，完整的要求

node-name@unit-address

unit-address一般都是外设寄存器的起始地址，有时候是I2C的设备地址，或者其他含义，具体节点具体分析。设备树里面常常遇到如下所示节点名字：

**intc: interrupt-controller@00a01000**

:前面是标签label，后面才是名字。intc，完整的名字是interrupt-controller@00a01000。&intc

# 四、创建小型的设备树模板

# 五、设备树在系统中的体现

系统启动以后可以在根文件系统里面看到设备树的节点信息。在/proc/device-tree/目录下存放着设备树信息。

内核启动的时候会解析设备树，然后在/proc/device-tree/目录下呈现出来。

# 六、特殊节点

1，aliases

2，chosen节点，主要目的就是将uboot里面bootargs环境变量值，传递给Linux内核作为命令行参数，cmd line。uboot里面bootargs值为：

bootargs=console=ttymxc0,115200 rw root=/dev/nfs nfsroot=192.168.1.249:/home/zzk/linux/nfs/rootfs ip=192.168.1.50:192.168.1.249:192.168.1.1:255.255.255.0::eth0:off

linux内核cmdline值为：

Kernel command line: console=ttymxc0,115200 rw root=/dev/nfs nfsroot=192.168.1.249:/home/zzk/linux/nfs/rootfs ip=192.168.1.50:192.168.1.249:192.168.1.1:255.255.255.0::eth0:off

uboot是如何向kernel传递bootargs。

经过查看发现chosen节点中包含bootargs属性，属性值和uboot的bootargs一致。

uboot接触过dtb，最终通过bootz 80800000 – 83000000 来启动内核。经过分析判断uboot拥有bootargs环境变量和dtb，因此最有可能“作案”。

最终发现在uboot的fdt\_chosen函数中会查找chosen节点，并且在里面添加bootargs属性，属性值为bootargs变量值。

# 七、特殊的属性

1、compatible属性，值是字符串。

根节点/下面的compatible。内核启动的时候会检查是否支持此平台，或机器。

不使用设备树的时候通过machine id来判断内核是否支持此机器。

#define MACHINE\_START(\_type,\_name) \

static const struct machine\_desc \_\_mach\_desc\_##\_type \

\_\_used \

\_\_attribute\_\_((\_\_section\_\_(".arch.info.init"))) = { \

.nr = MACH\_TYPE\_##\_type, \

.name = \_name,

#define MACHINE\_END \

};

MACHINE\_START(MX35\_3DS, "Freescale MX35PDK")

/\* Maintainer: Freescale Semiconductor, Inc \*/

.atag\_offset = 0x100,

.map\_io = mx35\_map\_io,

.init\_early = imx35\_init\_early,

.init\_irq = mx35\_init\_irq,

.init\_time = mx35pdk\_timer\_init,

.init\_machine = mx35\_3ds\_init,

.reserve = mx35\_3ds\_reserve,

.restart = mxc\_restart,

MACHINE\_END

展开以后：

static const struct machine\_desc \_\_mach\_desc\_MX35\_3DS \_\_used

\_\_attribute\_\_((\_\_section\_\_(".arch.info.init"))) = {

**.nr = MACH\_TYPE\_MX35\_3DS, //机器ID**

.name = "Freescale MX35PDK",

.atag\_offset = 0x100,

.map\_io = mx35\_map\_io,

.init\_early = imx35\_init\_early,

.init\_irq = mx35\_init\_irq,

.init\_time = mx35pdk\_timer\_init,

.init\_machine = mx35\_3ds\_init,

.reserve = mx35\_3ds\_reserve,

.restart = mxc\_restart,

};

使用设备树的时候不使用机器ID，而是使用根节点/下的compatible。

#define DT\_MACHINE\_START(\_name, \_namestr) \

static const struct machine\_desc \_\_mach\_desc\_##\_name \

\_\_used \

\_\_attribute\_\_((\_\_section\_\_(".arch.info.init"))) = { \

.nr = ~0, \

.name = \_namestr,

DT\_MACHINE\_START(IMX6UL, "Freescale i.MX6 Ultralite (Device Tree)")

.map\_io = imx6ul\_map\_io,

.init\_irq = imx6ul\_init\_irq,

.init\_machine = imx6ul\_init\_machine,

.init\_late = imx6ul\_init\_late,

.dt\_compat = imx6ul\_dt\_compat,

MACHINE\_END

展开以后就是：

static const struct machine\_desc \_\_mach\_desc\_IMX6UL \_\_used

\_\_attribute\_\_((\_\_section\_\_(".arch.info.init"))) = {

**.nr = ~0,**

.name = "Freescale i.MX6 Ultralite (Device Tree)",

.map\_io = imx6ul\_map\_io,

.init\_irq = imx6ul\_init\_irq,

.init\_machine = imx6ul\_init\_machine,

.init\_late = imx6ul\_init\_late,

**.dt\_compat = imx6ul\_dt\_compat,**

}；

# 八、Linux内核的OF操作函数

1、驱动如何获取到设备树中节点信息。在驱动中使用OF函数获取设备树属性内容。

2，驱动要想获取到设备树节点内容，首先要找到节点。