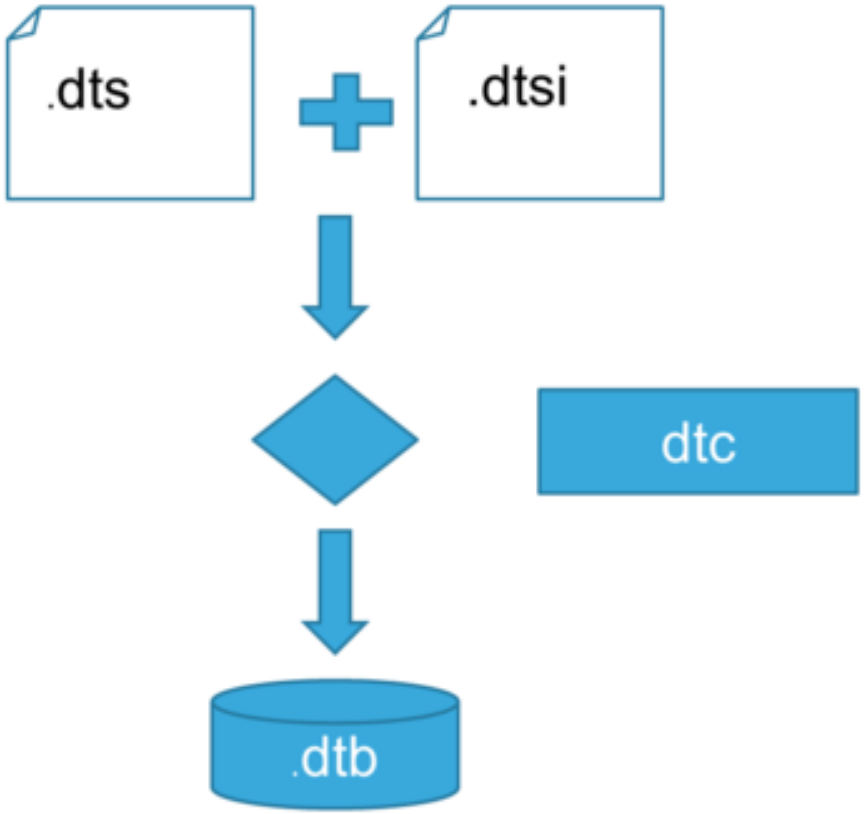


life.augmented

STM32MP1设备树

目标

#1 设备树概念

#2 修改STM32MP1设备树的方法与工具

#3 STM32MP1设备树修改示例

life.augmented

什么是设备树

- 设备树并不是从开始就存在，而是后来加入到Linux中
- 设备树主要用来描述系统的硬件结构
  - SoC的Identification
  - 运行时配置
  - 描述是系统不能动态获取（probe）的硬件信息
- Wiki参考
  - https://wiki.st.com/stm32mpu-ecosystem-v1/wiki/Device_tree

```
/ {
        …
model = "STMicroelectronics STM32MP157A-DK1 Discovery
Board";
        compatible = "st,stm32mp157a-dk1", "st,stm32mp157";

        aliases {
                ethernet0 = &ethernet0;
                serial0 = &uart4;
                serial1 = &usart3;
                serial2 = &uart7;
        };

        chosen {
                stdout-path = "serial0:115200n8";
        };
        …
          led {
                        compatible = "gpio-leds";
                        blue {
                                label = "heartbeat";
                                gpios = <&gpiod 11
          GPIO_ACTIVE_HIGH>;
                                linux,default-trigger = "heartbeat";
                                default-state = "off";
                        };
                };
                …
}
```

life.augmented

设备树规范

- 设备树有语法规范，但需要相应的驱动进行解释
  - 造成设备树的通用性与特殊性
- 设备树的语法结构
  - 树形结构
    - 根”/”没有parent
    - 每个节点只有一个parent
  - 节点
    - 每个节点一般代表系统中的一个设备
    - 包含子节点
    - 具有property-value对
  - 支持overlay

```
/ {                                                                    ---根节点
        …
model = "STMicroelectronics STM32MP157A-DK1 Discovery                  ---属性
Board";
    compatible = "st,stm32mp157a-dk1", "st,stm32mp157";                ---属性

    aliases {                                                          ---节点
        ethernet0 = &ethernet0;                                        ---属性
        serial0 = &uart4;                                              ---属性
        serial1 = &usart3;                                             ---属性
        serial2 = &uart7;                                              ---属性
    };

    chosen {                                                           ---节点
        stdout-path = "serial0:115200n8";                              ---属性
    };
            …
      led {                                                            ---节点
                compatible = "gpio-leds";                              ---属性
                blue {                                                 ---节点
                    label = "heartbeat";                               ---属性
                    gpios = <&gpiod 11 GPIO_ACTIVE_HIGH>;              属性
                    linux,default-trigger = "heartbeat";               ---属性
                    default-state = "off";                             ---属性
                };
            };
                …
}
```

语法参考： https://github.com/devicetree-org/devicetree-specification/releases/tag/v0.3

设备树的物理形式

- 源码形式
  - dts（device tree source ）
    - 文本格式
    - 一般为板级配置
  - dtsi
    - 文本格式
    - 一般为SOC配置
    - 可以被DTS包含的设备树文件
- 目标形式
  - dtb
    - 二进制文件格式
    - 通过编译工具进行编译

注：并不是只有dtsi才能被包含

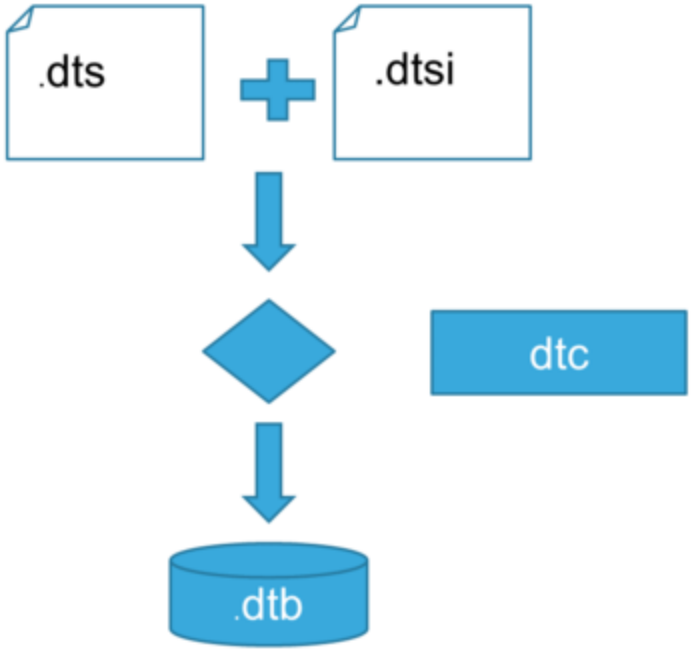
注：软件（例如内核）最终使用二进制设备树文件

注：二进制设备树文件不一定单独存在于文件系统

life.augmented

设备树的编译

- DTC(Device Tree Compiler) 将设备树源码编译成二进制文件
  - 输入文件： dts(内部包含dtsi以及其他dts)
  - 输出文件： dtb
- STM32MP1中设备树的编译
  - 在编译相应组件时自动生成
  - Linux设备树可单独生成
    - make dtbs



设备树的调试工具

- **在主机上调试设备树**
  - **dtc**
  - **fdtdump**
    - **从dtb输出可读的设备树**
  - **fdtget**
    - **从dtb读出一个值**
  - **fdtput**
    - **写入一个设备树的值**
- **在目标机上查看设备树**

示例:

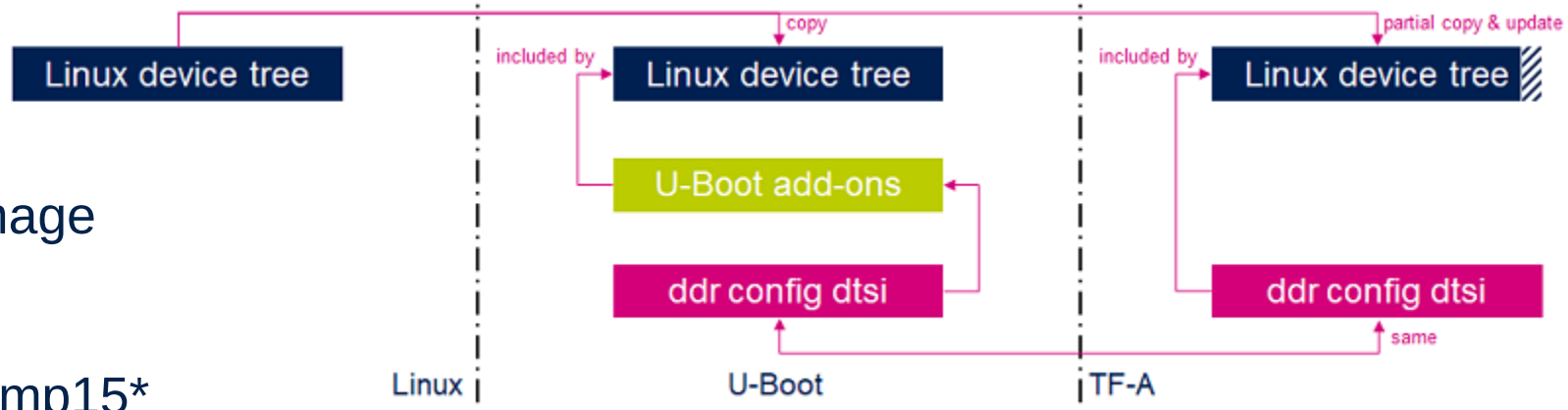
```
dtc -I dtb -O dts ../build/arch/arm/boot/dts/stm32mp157a-dk1.dtb

fdtdump ../build/arch/arm/boot/dts/stm32mp157a-dk1.dtb

fdtget ../build/arch/arm/boot/dts/stm32mp157a-dk1.dtb "/chosen" stdout-path

fdtput ../build/arch/arm/boot/dts/stm32mp157a-dk1.dtb "/chosen" stdout-path
"serial1:115200n8" -t s
```

```
root@stm32mp1:~# hexdump -C /proc/device-tree/model
00000000  53 54 4d 69 63 72 6f 65  6c 65 63 74 72 6f 6e 69  |STMicroelectroni|
00000010  63 73 20 53 54 4d 33 32  4d 50 31 35 37 41 2d 44  |cs STM32MP157A-D|
00000020  4b 31 20 44 69 73 63 6f  76 65 72 79 20 42 6f 61  |K1 Discovery Boa|
00000030  72 64 00                                          |rd.|
00000033
```

life.augmented

STM32MP1中的设备树

- TF-A
  - 源码路径：fdts/stm32mp15*
  - 输出：与最终TF-A编译成一个image
- U-Boot
  - 源码路径： arch/arm/dts/stm32mp15*
  - 输出：与最终U-Boot编译成一个image
- Linux Kernel
  - 源码路径： arch/arm/boot/dts/stm32mp15*
  - 输出： 可单独存放在文件系统里



https://wiki.st.com/stm32mpu/wiki/STM32MP15_device_tree

注意：在STM32MP1 ecosystem 2.0之前并没有按型号的设备树，但不影响使用。

- 例如:STM32MP151

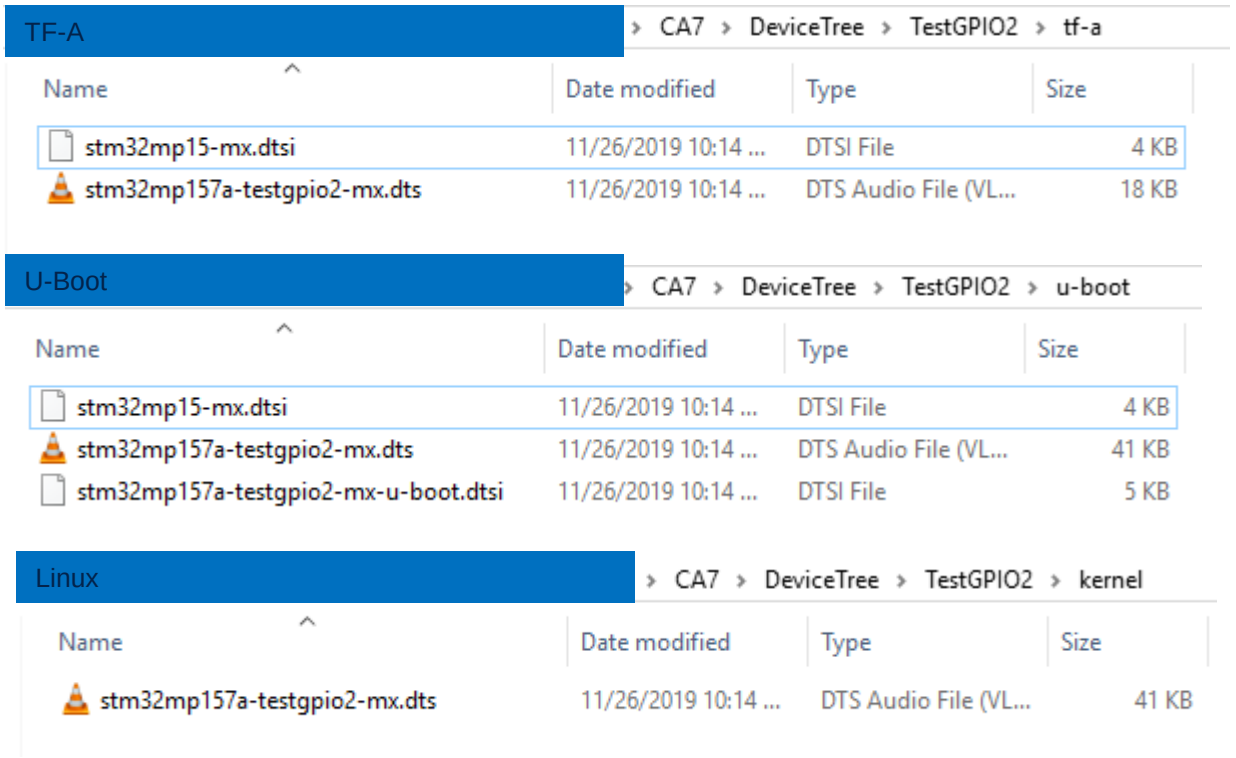
life.augmented

修改编写STM32MP1设备树

- STM32MP1开发中的设备树修改需求
  - 文件重命名
    - 对UBoot设备树重命名要考虑Makefile自动包含规则
  - 对系统进行描述配置
    - 包括DDR
- 工具
  - 使用文本编辑工具编辑源码
  - 使用原有编译系统进行编译

STM32CubeMX辅助修改设备树

- 使用STM32CubeMX辅助生成设备树
  - 可直接在Windows下运行
  - 生成板级配置
    - 不包含SOC配置
  - 生成DDR配置
    - STM32CubeMX还支持DDR Tuning以及测试
  - 不生成非STM32MP1的设备树
  - 使用copy&paste进行合并



STM32MP1设备树参考资料

- STM32MP1 Wiki
  - 地址
    - https://wiki.st.com/stm32mpu/wiki/Category:Device_tree_configuration
  - 包含大量经过验证的设备树配置范例
  - 包含一些在STM32CubeMX还不支持的用法
- STM32MP1参考手册
- STM32MP1源代码
- STM32MP1 Linux DT Binding文件

life.augmented

应用实例-Secure/Non-Secure

- 需求
  - RCC有些寄存器在内核中写入报错
- 原理
  - 查看参考手册RM0436以及源代码
    - 默认RCC是配置在Secure World
    - 将RCC放在Non-Secure World
- 步骤
  - 修改板级TF-A设备树中rcc的secure-status
  - 重新编译TF-A
  - 重新烧录TF-A

RM0436

When TZEN = '1', some RCC and PWR registers can only be modified using secure accesses. The reading of registers is always allowed.

TF-A:Stm32mp1_clk.c

```
/* Check status field to disable security */
if (!fdt_get_rcc_secure_status()) {
    mmio_write_32(rcc_base + RCC_TZCR, 0);
}
```

```
diff --git a/fdts/stm32mp157a-dk1.dts b/fdts/stm32mp157a-dk1.dts
index a8f91a3..5413165 100644
--- a/fdts/stm32mp157a-dk1.dts
+++ b/fdts/stm32mp157a-dk1.dts
@@ -217,7 +217,7 @@

 /* CLOCK init */
 &rcc {
-       secure-status = "okay";
+       secure-status = "disabled";
        st,hsi-cal;
        st,csi-cal;
        st,cal-sec = <60>;
```

应用实例-增加一个USART

- 需求
  - 在内核中增加一个USART
- 原理
  - gpio controller node 已经提供在 SoC dtsi中
  - 需要将相应的USART使能
- 步骤
  - 修改板级配置
  - 重新编译设备树
  - 复制设备树到目标机的boot目录

参考资源

- Wiki
  - https://wiki.st.com/stm32mpu/wiki/Pinctrl_device_tree_configuration
  - https://wiki.st.com/stm32mpu/wiki/GPIO_device_tree_configuration
- 内核文件St,stm32-pinctrl.txt
  documentation\devicetree\bindings\pinctrl

```
diff --git a/arch/arm/boot/dts/stm32mp157a-dk1.dts b/arch/arm/boot/dts/stm32mp157a-dk1.dts
index 9ec45de..9a812af 100644
--- a/arch/arm/boot/dts/stm32mp157a-dk1.dts
+++ b/arch/arm/boot/dts/stm32mp157a-dk1.dts
@@ -730,7 +730,7 @@
        pinctrl-1 = <&usart3_sleep_pins_b>;
        pinctrl-2 = <&usart3_idle_pins_b>;
        st,hw-flow-ctrl;
-       status = "disabled";
+       status = "okay";
 };
```

life.augmented

应用实例-三USB

- 需求
  - 让STM32MP1支持三个USB
- 原理
  - STM32MP1可同时支持
    - 两个高速USB 2.0
    - 一个全速USB 2.0
  - STM32CubeMX目前支持两个高速USB2.0
- 步骤
  - 修改OTG支持全速
  - 编译设备树
  - 复制设备树到目标机的boot目录

- 参考资源
  - OTG
    - https://wiki.st.com/stm32mpu/wiki/OTG_device_tree_configuration
  - USBH
    - https://wiki.st.com/stm32mpu-ecosystem-v1/wiki/USBH_device_tree_configuration

```
&usbotg_hs {
        compatible = "st,stm32mp1-fsotg", "snps,dwc2";
        pinctrl-names = "default";
        pinctrl-0 = <&usbotg_hs_pins_a
&usbotg_fs_dp_dm_pins_a>;
        vbus-supply = <&vbus_otg>;
        status = "okay";
};
```

life.augmented

测验

- 如何修改设备树将RCC改在 non-secure?
- 如何修改设备树增加一个USART?
- 如何修改设备树使用三个USB?

Thank you

life.augmented