

Embedded Linux Porting (C7)

Linux Device Tree

Agenda



- Understanding Device Tree Structure
- Nodes in DTS
- - Properties of Nodes
- Kernel API's to process device tree data
- Compiling Device Tree and Flashing

Why Device Tree



- 1. Kernel maintainers wanted unified way to add more board without adding board.c files which are hard to maintain
- 2. Change in hardware configuration should be possible without re-compiling the kernel
- 3. Re-use same kernel image for different boards of same SOC

- 1. Board File for each board under mach folder
 - -> linux/arch/arm/mach-<soc>/board.c
- 2. Device Tree replaced with generic board file, so only one generic board file with different dts files defining different boards.
 - -> linux/arch/arm/mach-<soc>/board-generic.c
 - -> linux/arch/arm/boot/dts/board.dts

Linux Device Driver Model



- 1. Platform Device
- 2. Platform Data
- 3. Platform Driver

Device Tree Node



```
Node name
                             Unit address
                                       Property name
                                                        Property value
                node@0 {
                     a-string-property = "A string";
                    a-string-list-property = "first string", "second string";
Properties of node@0
                    a-byte-data-property = [0x01 0x23 0x34 0x56];
                     child-node@0 {
                         first-child-property;
                                                            - Bytestring
                         second-child-property = <1>;
                         a-reference-to-something = <&node1>;
                    };
                                              A phandle_
                     child-node@1 {
                                              (reference to another node)
                    };
         Label
                node1: node@1 {
                     an-empty-property;
                     a-cell-property = <1 2 3 4>;
                     child-node@0 {
                                                 Four cells (32 bits values)
                    };
                };
            };
```

Device Tree Node



```
auart0: serial@8006a000 {
          Defines the "programming model" for the device. Allows the
          operating system to identify the corresponding device driver.
          compatible = "fsl,imx28-auart", "fsl,imx23-auart";
          Address and length of the register area.
          reg = <0x8006a000 0x2000>;
          Interrupt number.
          interrupts = <112>;
          DMA engine and channels, with names.
         dmas = <\&dma_apbx 8>, <\&dma_apbx 9>;
         dma-names = "rx", "tx";
          Reference to the clock.
         clocks = <\&clks 45>;
          The device is not enabled.
         status = "disabled";
```

Device Tree Functions



- 1. of_platform_populate()
- of_driver_match_device()
- 3. of_find_node_by_name()
- 4. of_property_read_u32()

Functions defined in:

linux/include/linux/of.h
linux/include/linux/of_platform.h
linux/include/linux/of_device.h

Compiling Device Tree



#To compile device Tree from Kernel Source \$make dtbs

#To compile manually using dtc \$dtc -I dts -O dtb <dts_filename> -o <dtb_filename>

#To extract dts file from dtb file \$dtc -I dtd -O dts <dtb_filename> -o <dts_filename>

DeviceTree Experiments



1. Delete the DT & try to boot the board (capture the error log)



Open Discussions











Developer Wiki







Attribution 4.0 International (CC BY 4.0)

This is a human-readable summary of (and not a substitute for) the license. Disclaimer.

You are free to:

Share — copy and redistribute the material in any medium or format



Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.